VisualAge Pacbase 2.5

Pacbench Client / Serveur

Guide Utilisateur – Volume III

Clients Graphiques

DDOVA000255F

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.

Sommaire

A la suite de ce sommaire, vous disposez d'une Table des Matières détaillée.

Table des Matières

Préambule

Contenu du Volume

L'objet de ce manuel est de guider le développeur dans la réalisation d'applications graphiques clientes dans Pacbench Client/Serveur. La partie serveur de ces applications est générée par les Services Applicatifs de VisualAge Pacbase. La Station VisualAge est utilisée pour développer des clients en langage Java ou Smalltalk exécutables sur micro-ordinateur ou accessibles depuis le Web. Les clients graphiques peuvent également être développés par un outil utilisant la technologie COM et, pour un client Java, par tout outil de développement Java supportant la version 1.1 du JDK.

- L'introduction décrit brièvement les étapes du développement côté client graphique, de la génération des éléments extraits du Référentiel à la construction graphique des applications.
- Puis la génération des objets Proxy est décrite en détail pour chaque type de client (Java, Smalltalk puis COM) dans un chapitre spécifique.
- Vient ensuite pour chaque type d'environnement de développement, un chapitre qui expose les principes généraux de l'utilisation des éléments de l'interface publique des composants générés, présente la gestion des erreurs ainsi que la gestion de la communication. Les chapitres consacrés aux clients Java et Smalltalk présentent en outre des exemples commentés pas à pas de construction graphique d'applications GUI standard et Web et fournissent des informations sur le test et le packaging.
- Puis, vous trouverez un chapitre consacré au Pont Pacbench Client/Serveur. Dans sa version Java, le Pont permet de sauvegarder les références croisées des composants graphiques dans le Référentiel VisualAge Pacbase. Dans la version Smalltalk, il permet de sauvegarder le source et les références croisées des composants graphiques dans le Référentiel et de les restaurer ensuite dans la Station VisualAge.
- Enfin, l'Annexe 1 présente le paramétrage des logiciels externes et l'Annexe 2 donne des indications sur la personnalisation du middleware.

Prérequis

Avant de lire ce volume, vous devez impérativement connaître les grands principes du développement d'applications Client/Serveur avec VisualAge Pacbase. Ils sont supposés connus dans le présent Volume. Consultez le *Guide Utilisateur Pacbench C/S, Vol. I : Concepts - Architectures - Environnements*.

Le *Guide Utilisateur Pacbench C/S, Vol. II – Services Applicatifs* présente le développement des composants serveurs que vous pourrez ensuite intégrer dans l'application cliente sous forme d'objets Proxy. La lecture approfondie de ce Volume n'est pas impérative pour le développement des clients graphiques, mais elle permet néanmoins de mieux comprendre ce que sont les objets Proxy.

Ces manuels ne sauraient toutefois contenir toutes les informations relatives au développement d'applications Client/Serveur avec VisualAge Pacbase, c'est pourquoi vous aurez également besoin de consulter notamment les manuels suivants :

- Manuel de Référence *Pacbench Client/Serveur - Interface Publique des Composants Générés*,
- Manuel de référence *Station de Travail VisualAge Pacbase*,
- Manuel d'exploitation du *Pont Pacbench Client/Serveur* spécifique à votre plateforme,
- Documentation associée à l'utilisation de la Station VisualAge Smalltalk, notamment :
  - *VisualAge Smalltalk - Getting Started*,
  - *VisualAge Smalltalk - User's Guide*,
  - *VisualAge Smalltalk - User's Reference*,
  - *VisualAge Web Connection - User's Guide*,
  - *IBM Smalltalk: Introduction to Object-Oriented Programming with IBM Smalltalk*,
  - *IBM Smalltalk: Programmer's Reference*,
  - *IBM Smalltalk User's Guide*.
- Documentation associée à l'utilisation de la Station VisualAge Java, notamment la documentation HTML en ligne ou celle associée à tout autre outil Java utilisé.
- Documentation associée à votre environnement de développement COM.

Conventions typographiques

La police `Courier New` est utilisée pour toute chaîne de caractères à saisir, affichée ou correspondant à du code généré.

Les titres des Manuels ainsi que les titres des chapitres dans les renvois sont en italique.

Les symboles suivants sont utilisés :

☞ note, remarque, précision importante

renvoi à un autre emplacement dans la documentation

truc ou astuce, précision utile

manipulation à effectuer dans un Outil ou un Editeur

précaution à prendre (manipulation risquée ou irréversible…)

Conventions terminologiques

- Une PVD désigne une Proxy Vue de Dossier.
- Une PVL désigne une Proxy Vue Logique.
- Les termes objet(s) Proxy et composant(s) Proxy désignent de manière générique les Proxy Vues de Dossier, Proxy Elémentaires et Proxy Vues Logiques.

1. Introduction

Avec la fonctionnalité Clients Graphiques de Pacbench Client/Serveur, vous avez la possibilité de développer des applications graphiques dans VisualAge en langage Java ou Smalltalk mais aussi dans un environnement de développement utilisant les spécifications COM de Microsoft.

Avec la Station VisualAge, outre les applications graphiques standard qui seront exécutées sur micro-ordinateur, il est possible de réaliser des applications accessibles depuis le Web (Intranet ou Internet).

Quelque soit votre environnement de développement, le principe est basé sur l'utilisation d'un composant proxy généré qui, une fois importé, permet de commander à distance les services du ou des Composant(s) Applicatifs correspondants.

Pacbench C/S vous permet donc d'intégrer harmonieusement des serveurs développés avec la fonctionnalité Services Applicatifs avec des clients développés dans la Station VisualAge ou un environnement adapté au standard COM. Pacbench C/S fournit en effet tous les éléments nécessaires à l'interfaçage entre les serveurs et les clients.

☞ Pour des informations sur le Composant Applicatif, reportez-vous au *Guide Utilisateur Pacbench C/S, Vol. II : Services Applicatifs*.

Il est également important de proposer un lieu de stockage unique pour bénéficier d'une intégration assurant des mécanismes globaux, notamment la réutilisation et les références croisées. Le stockage dans le Référentiel VisualAge Pacbase permet également de bénéficier de la gestion des versions et des procédures de sécurité (archivage, sauvegarde, réorganisation).

Le Référentiel est donc le lieu de stockage de référence des clients graphiques développés dans VisualAge.

Tous les objets manipulés dans VisualAge peuvent être stockés dans le Référentiel et restitués ensuite (dans le cas de Smalltalk) dans la Station VisualAge. C'est le principe du Pont Pacbench Client/Serveur.

1.1. Les étapes du développement

Le développement d'une application graphique dans Pacbench C/S comprend le développement des composants Serveur puis celui des composants Clients.

☞ Le développement des serveurs - comprenant le développement des Composants Applicatifs, éventuellement celui du Dossier et de la Vue de Dossier dans la Station VisualAge Pacbase, l'extraction des Composants Applicatifs (pour un développement mono-vue) ou celle de la Vue de Dossier – est documenté dans le *Guide Utilisateur Pacbench C/S, Vol. II – Services Applicatifs*.

Le développement côté Client comprend :

- la génération/Importation des objets Proxy

  La phase de génération/importation produit les classes dont les objets seront utilisables par les applications clientes à développer dans la station Client. Elle est effectuée en local par un générateur piloté par une interface graphique.

  Ce générateur prend en entrée le résultat d'extraction intialement transféré sur le poste de développement client. Les éléments constitutifs de la Vue de Dossier ou du/des Composant(s) Applicatif(s), sont alors regroupés dans l'objet Proxy. Tous les éléments nécessaires à l'appel des Composants Applicatifs associés seront alors présents et initialisés dans l'application cliente.

Pour plus de détails sur la génération, consultez le chapitre **2** pour un Client Java, le chapitre **3** pour un Client Smalltalk ou le chapitre **4** pour un Client au standard COM.

- la réalisation de l'application cliente

  Le développement de l'application cliente inclut l'intégration des objets Proxy ainsi que l'appel de leurs services.

Vous trouverez des informations sur les actions correspondant à ces services dans la section **5.1.3** pour un Client Java, **6.1.3** pour un Client Smalltalk ou **7.1.1.3** pour un Client au standard COM.

Vous trouverez aussi un exemple commenté de développement d'une application graphique standard et Web dans les sous-chapitres **5.2** et **5.3** pour un Client Java, **6.2** et **6.3** pour un Client Smalltalk.

Le développement de l'application cliente inclut également la construction graphique de l'application basée sur les outils standard de votre station de développement et éventuellement l'écriture de code. Ces sujets ne relèvent pas de ce manuel, reportez-vous à la documentation VisualAge ou celle associée à tout autre outil de développement Client.

1.2. Compatibilité des Composants Applicatifs / Objets Proxy

Les Composants Applicatifs et les objets Proxy doivent être générés avec le même numéro de version car une différence de version peut provoquer des déphasages qui se traduisent par des incohérences dans l'application cliente.

Le numéro de version est différent si :

- vous avez regénéré le Composant Applicatif en modifiant sa version sans générer l'objet Proxy associé,
- ou l'application graphique générée contenant le nouvel objet Proxy n'a pas été mise en exploitation dans VisualAge,
- ou le Composant Applicatif généré n'a pas été mis en exploitation.

Pour éviter d'éventuels déphasages, vous devez implémenter le contrôle des versions en positionnant une option dans le Composant Applicatif. Cette option envoie une erreur si un déphasage de numéro de version est détecté lors de l'appel du Client VisualAge au Composant Applicatif.

L'option qui gère les numéros de version est l'option **NUVERS**. Pour plus de détails, reportez-vous au *Guide Utilisateur Pacbench C/S, Vol. II : Services Applicatifs*.

2. Génération VisualAge Java

Pour développer un Client Pacbench C/S dans VisualAge, vous utilisez des classes générées mais également un grand nombre de classes génériques qui sont livrées avec le produit pour éviter la multiplication des éléments à chaque nouvelle génération. Contrairement aux classes générées documentées plus loin, les classes génériques ne dépendent pas des caractéristiques de la Vue Logique traitée.

2.1. Classes génériques livrées

Vous devez vous assurer que les classes génériques sont installées sur votre station avant de commencer tout développement.

Les classes génériques Pacbench C/S sont livrées dans les 3 packages suivants :

- **`com.ibm.vap.generic`**

Ce package contient :

- les classes ancêtres des classes **`ProxyLv`** générées, c'est-à-dire les classes suivantes :
  - **`ProxyLv`**
  - **`HierarchicalProxyLv`**
  - **`DependentProxyLv`**
  - **`Folder`**
  - **`ReferenceProxyLv`**
- les classes **`Data`** génériques (ancêtres des classes **`selectionCriteria`** et **`DataDescription`** générées)
- les classes du cache local, c'est-à-dire les classes suivantes :
  - **`Node`**
  - **`HierarchicalNode`**
  - **`DependentNode`**
  - **`RootNode`**
  - **`ReferenceNode`**
- les exceptions et erreurs Pacbench C/S, c'est-à-dire les classes suivantes :
  - **`VapException`**
  - **`LocalException`**
  - **`ServerException`**
  - **`CommunicationError`**
  - **`SystemError`**
- les classes de présentation sous forme de liste des classes **`DataDescription`** générées (manipulées par la propriété **`rows`** des objets Proxy).
- les classes décrivant les propriétés des objets Proxy telles qu'elles sont définies dans VisualAge Pacbase :
  - **`VapProxyProperties`**
  - **`VapHierarchicalProxyProperties`**
  - **`VapDependentProxyProperties`**

- **VapFolderProperties**
- **VapReferenceProxyProperties**

- **com.ibm.vap.exchange**

Ce package contient les classes de gestion du Gestionnaire d'échanges.

2.1.1. Documentation en ligne des classes génériques

Une documentation au format HTML est livrée avec le runtime Pacbench C/S. Cette documentation porte sur :

- les classes génériques, ancêtres des classes d'exécution **ProxyLv** et **data** générées,
- les erreurs et exceptions levées par ces classes d'exécution,
- les beans associés aux propriétés de type Rubriques VA Pac, utilisés dans le Composition Editor pour le maquettage rapide de ces données.

  Dans la palette **awt**, ces beans sont les suivants :
  - **Pacbase Text Field**
  - **Pacbase Integer Field**
  - **Pacbase Decimal Field**
  - **Pacbase Date Field**
  - **Pacbase Time Field**
  - **Pacbase Long Field**
  - **Pacbase Text Choice**
  - **Pacbase Integer Choice**
  - **Pacbase Decimal Choice**
  - **Pacbase Date Choice**
  - **Pacbase Time Choice**
  - **Pacbase Long Choice**

  Dans la palette **swing**, ces beans sont les suivants :
  - **Pacbase Swing Text Field**
  - **Pacbase Swing Integer Field**
  - **Pacbase Swing Decimal Field**
  - **Pacbase Swing Date Field**
  - **Pacbase Swing Time Field**
  - **Pacbase Swing Long Field**
  - **Pacbase Swing Text ComboBox**
  - **Pacbase Swing Integer ComboBox**
  - **Pacbase Swing Decimal ComboBox**
  - **Pacbase Swing Date ComboBox**
  - **Pacbase Swing Time ComboBox**
  - **Pacbase Swing Long ComboBox**
  - **Pacbase Swing Date RadioButtonGroup**
  - **Pacbase Swing Decimal RadioButtonGroup**
  - **Pacbase Swing Integer RadioButtonGroup**

2.2. Lancement du générateur

Pour lancer le générateur :

- dans une session DOS ou OS/2, exécutez la commande **`java com.ibm.vap.generator.view.Main [-lang [fr] [en]]`**

  ☞ Pour utiliser cette commande, le JDK (Java Developer ToolKit) doit être installé et la commande **`java`** doit être déclarée dans le **`PATH`**.

  L'option **`lang`** permet de choisir une langue pour la présentation de l'interface graphique du générateur : **`fr`** pour français, **`en`** pour English.

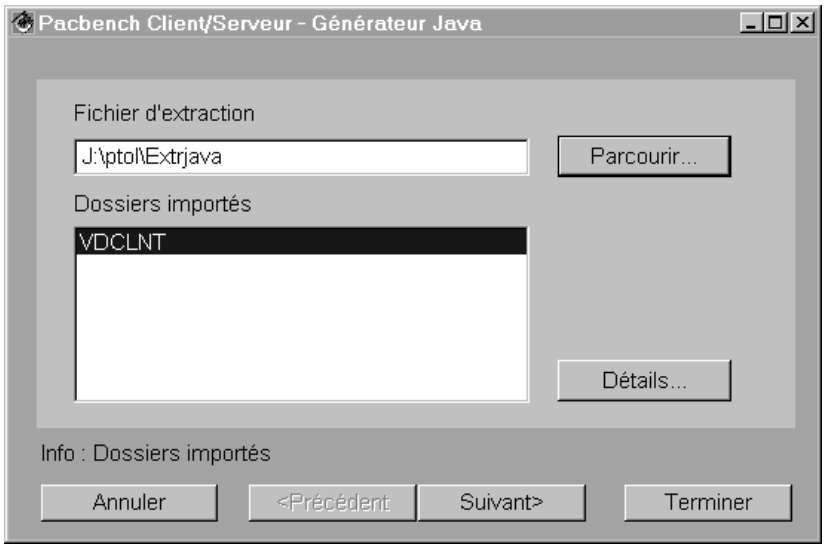
  Par défaut, la langue du système est utilisée.

  Le générateur dispose également des options de génération suivantes :
  - **`-input INPUT_FILE`** : cette option permet de spécifier le fichier d'extraction.
  - **`-output OUTPUT_DIR`** : cette option permet de spécifier le répertoire dans lequel les classes générées seront stockées.
  - **`-proxy PROXY_PACKAGE`** : cette option permet de spécifier le package Java dans lequel les classes correspondant aux composants Proxy seront générées.
  - **`-data DATA_PACKAGE`** : cette option permet de spécifier le package Java dans lequel les classes de données seront générées.
  - **`-beans`** : cette option permet de générer des classes dont les objets sont conformes aux spécifications des composants JavaBeans.
  - **`-swing`** : cette option permet de générer toutes les informations nécéssaires à l'exploitation des classes visuelles Swing.
  - **`-eab`** : cette option permet de générer des classes capables de s'interfacer avec les classes EAB d'IBM.

  ☞ Ces options correspondent toutes à des paramètres de génération proposés dans l'interface graphique du générateur ; pour plus de détails, voir la section 2.3 *Mode de fonctionnement*.

- ou, à partir du Gestionnaire de fichiers Windows ou l'Explorateur Windows NT, lancez le programme **`vapgen.exe`**.
- si le générateur a été préalablement importé dans la Station VisualAge :
  - dans le Workbench, sélectionnez le Projet dans lequel le générateur a été importé.
  - pour choisir la langue du générateur, sélectionnez le package **`com.ibm.vap.generator.view`** puis la classe **`Main`**. Affichez les propriétés de cette classe. Dans la fenêtre qui s'ouvre, dans la zone **`Command Line Argument`**, saisissez l'option langue souhaitée comme suit : **`-lang fr`** ou **`–lang en`**.
    Si la zone est à blanc, l'interface du générateur sera en anglais.
  - pour lancer le générateur, affichez le menu contextuel du projet. Sélectionnez le sous-menu **`Run`**, le choix **`Run Main…`** puis **`Main`**, ou cliquez sur l'icône **`Run`** dans la barre d'outils et sélectionner **`Main`**.

Le formulaire principal du générateur s'ouvre :



2.3. Mode de fonctionnement

- **Sélection des Vues de Dossier à générer**

Vous devez d'abord renseigner le chemin d'accès au fichier d'extraction dans la zone **`Fichier d'extraction`**. Vous pouvez :

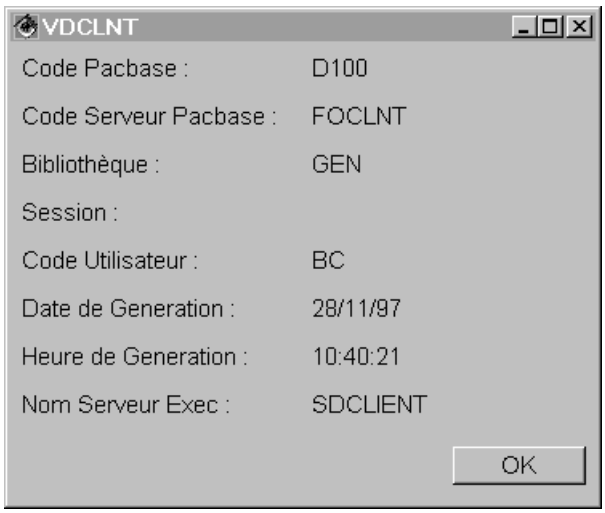
- le saisir dans la boîte edit, ou
- cliquer sur le bouton **`Parcourir...`**. Une fenêtre s'ouvre alors dans laquelle vous pouvez sélectionner le chemin complet du fichier à l'aide des différentes boîtes de liste.

Une fois le nom de fichier saisi, la liste de la ou des Vue(s) de Dossier contenue(s) dans le fichier d'extraction s'affiche dans la zone **`Dossiers importés`**.

Lorsqu'il existe plusieurs Vues de Dossier dans la liste, elles sont toutes automatiquement sélectionnées. Le cas échéant, désélectionnez celle(s) que vous ne souhaitez pas générer.

☞ Le message **`Dossiers importés`** indique que le fichier d'extraction est correct et que les Vues de Dossier qu'il contient sont identifiées et prêtes pour la génération.

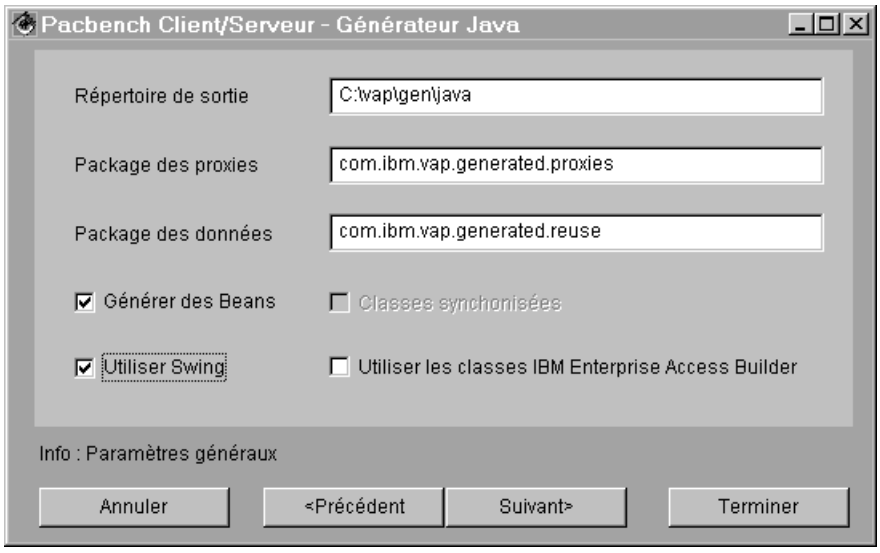
- **Autres fonctionnalités du formulaire**
  - Le bouton **Détails…** permet d'afficher le contexte de génération **GVC** pour la Vue de Dossier sélectionnée.



  - Le bouton **Suivant** provoque une navigation vers le formulaire qui vous permettra de spécifier les paramètres de génération. Pour des détails sur ces paramètres par défaut, reportez-vous au paragraphe suivant.
  - Si vous souhaitez générer immédiatement toutes les Vues de Dossier sélectionnées, en utilisant les paramètres par défaut, cliquez sur le bouton **Terminer**. Une boîte de message vous demande de confirmer la création du répertoire de sortie s'il n'existe pas encore. Cliquez sur OK ; la génération est lancée.

- **Options de génération**

Si vous avez cliqué sur **Suivant** dans le formulaire principal, le formulaire suivant s'affiche :



Ce formulaire vous permet de spécifier les paramètres de génération. Le formulaire présente ici les paramètres proposés par défaut.

- Le champ **Répertoire de sortie** contient le chemin d'accès au répertoire dans lequel les classes seront générées.
- Le champ **Package des proxies** contient le nom du package Java dans lequel les classes correspondant aux composants Proxy seront générées.
- Le champ **Package des données** contient le nom du package Java dans lequel les classes de données seront générées.
- L'option **Générer des beans** permet de spécifier la création de beans ou de classes Java standard.
  - ☞ Cette option permet de générer des classes dont les objets sont conformes aux spécifications des JavaBeans, utilisables en programmation visuelle. Voir également le sous-chapitre **2.5**, *Résultats de la génération*.
- L'option **Classes synchronisées** permet de spécifier la création de classes supportant le multithreading.
  - ☞ Par exemple, des composants Proxy pourront être simultanément appelés par plusieurs programmes ou *threads*.
- L'option **Utiliser Swing** permet de générer toutes les informations nécessaires à l'exploitation des classes graphiques Swing intégrées à VisualAge Java à partir de la version 2.
  - ☞ Il sera possible de connecter dans le Visual Composition Editor la propriété **rows** des composants Proxy aux données d'une JTable (composant Swing comportant plusieurs lignes et plusieurs colonnes).
- Si l'option **Utiliser les classes IBM Enterprise Access Builder** est cochée, le générateur crée des classes pouvant s'interfacer avec les classes EAB d'IBM.
  - ☞ Il sera notamment possible de connecter dans le Visual Composition Editor la propriété **rows** des composants Proxy aux données d'un container EAB.

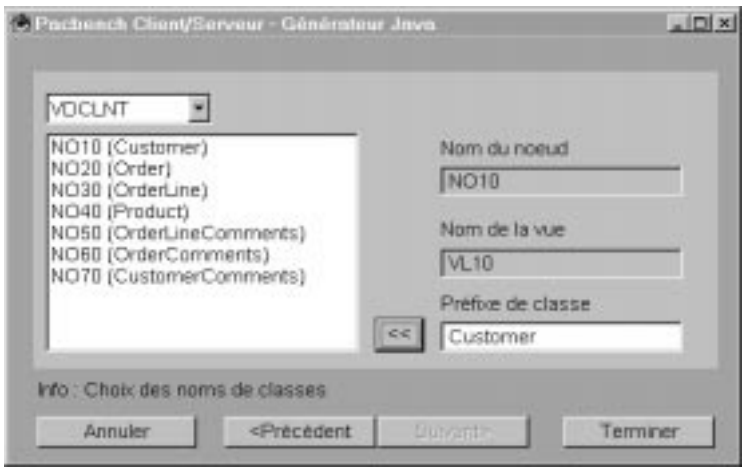
   Cette option génère des erreurs en cas d'utilisation avec VisualAge Java à partir de la version 2.

Ce formulaire contient également les boutons-poussoirs suivants :

- Le bouton **Précédent** provoque la navigation vers le formulaire principal.
- Le bouton **Suivant** provoque la navigation vers le formulaire de modification des préfixes des classes générées.
- Si vous souhaitez générer immédiatement toutes les Vues de Dossier sélectionnées, en utilisant les noms de classes par défaut, cliquez sur le bouton **Terminer**. Une boîte de message vous demande de confirmer la création du répertoire de sortie s'il n'existe pas déjà. Cliquez sur OK ; la génération est lancée.

- **Modification des préfixes des classes générées**

Si vous avez cliqué sur **Suivant** dans le formulaire des paramètres de génération, le formulaire suivant s'affiche :



Ce formulaire vous permet de modifier les préfixes des classes générées lorsque les préfixes issus du serveur ne vous conviennent pas, parce qu'ils ne sont pas suffisamment explicites par exemple. Par défaut, le préfixe est constitué par le nom en clair de la Vue Logique associée au nœud.

La liste déroulante contient la liste de toutes les Vues de Dossier que vous avez sélectionnées en vue de leur génération dans le formulaire principal.

- La zone de liste contient tous les nœuds composant la Vue de Dossier sélectionnée dans la liste déroulante. Chaque nœud est identifié par son code et le nom en clair de la Vue Logique (préfixe).
- Pour chaque nœud sélectionné dans la liste, la partie droite du formulaire rappelle le code du nœud et le code de la Vue Logique associée.
- Pour modifier le préfixe d'un nœud, procédez comme suit :
  - sélectionnez le nœud dans la liste,
  - entrez un nouveau préfixe dans le champ **Préfixe de classe**,
  - cliquez sur  pour affecter le préfixe au nœud, ou appuyez sur la touche **Entrée** pour affecter le préfixe au nœud et sélectionner automatiquement le nœud suivant dans la liste.

EXEMPLE :

Si le préfixe d'un nœud vaut **Customer**, les classes générées pour ce nœud sont codifiées de la manière suivante :

- **CustomerProxyLv**
- **CustomerDataDescription**
- **CustomerSelectionCriteria**
- etc.

Vous devez vous assurer de l'unicité et de la validité des noms de classes.

Ce formulaire contient également les boutons suivants :

- Le bouton **Précédent** provoque la navigation vers le formulaire précédent.
- Le bouton **Terminer** lance la génération. Une boîte de message vous demande de confirmer la création du répertoire de sortie s'il n'existe pas déjà. Cliquez sur OK ; la génération est lancée.

2.4. Import des classes générées dans VisualAge

Pour importer les classes générées dans VisualAge, procédez comme suit :

- Depuis le Workbench de VisualAge, sélectionnez le choix **Import…** dans le menu **File**.
- Dans la fenêtre **SmartGuide – Type of import**, entrez le nom du projet dans lequel les classes seront importées ou sélectionnez un projet existant à l'aide du bouton **Browse**.
  - ☞ Il est conseillé de ne pas importer les classes générées dans l'un des projets VisualAge Pacbase ou Java standard.
- Pour le type d'import, sélectionnez l'option **Entire Directory (Including resources)**.
  - ☞ L'option **Java files** est également possible. Dans ce cas, les icônes spécifiques à Pacbench Client/Serveur ne seront pas disponibles.
- Cliquez sur **Next** pour accéder au formulaire suivant. Sélectionnez le répertoire dans lequel les classes ont été générées à l'aide du bouton **Browse**.
  - ☞ Ce répertoire est celui que vous avez indiqué lors de la génération dans le champ **Répertoire de sortie**, dans le formulaire réservé au positionnement des options de génération : dans notre exemple, il s'agit de **C:\vap\gen\java**.

2.5. Résultats de la génération

Les éléments générés dépendent toujours du type de service effectué par le Composant Applicatif. Ils ne seront pas les mêmes selon que le Composant Applicatif effectue des services de mise à jour ou simplement de lecture.

Le fichier généré contient uniquement les classes qui dépendent des caractéristiques de la Vue Logique traitée.

2.5.1. Introduction

A l'issue de la génération, les fichiers suivants sont créés :

- Le fichier des localisations, **VAPLOCAT.INI** est créé dans le répertoire de sortie de génération.

Ce fichier doit être enrichi manuellement. Pour des informations sur la manière de procéder, reportez-vous à la section 5.5.3.

Ce fichier doit être fourni en entrée de la gateway.

- Les fichiers sources des classes générées et ressources nécessaires à leur édition.
  - ☞ Si vous avez coché l'option **Générer des beans**, une classe **BeanInfo** est générée pour chaque classe d'exécution (**ProxyLv** et **data**). La classe **BeanInfo** associée à une classe donnée (bean) porte le même nom que ladite classe, suivi du suffixe **BeanInfo**.

    Par exemple, lorsqu'une classe **CustomerProxyLv** (Proxy Racine dans notre exemple) est générée, une classe **CustomerProxyLvBeanInfo** est systématiquement générée.

Une classe **BeanInfo** contient des informations d'édition (libellé, icône, etc.) pour la classe d'exécution associée. Elle contient les méthodes publiques qui fournissent des informations sur la classe du bean associé, comme le nom de classe, les propriétés, méthodes et événements exposés par le bean.

2.5.2. Classes générées

Dans VisualAge, les éléments générés par Pacbench Client/Serveur correspondent à des classes dont la codification est composée d'un préfixe et d'une partie codée en dur dans le générateur. Le préfixe correspond au nom en clair de la Vue Logique qui comporte 36 caractères au maximum.

Si le préfixe originaire du serveur ne vous convient pas, vous avez la possibilité de le modifier lors de la génération. Pour plus de détails, reportez-vous au point *Modification des préfixes* des classes générées du sous-chapitre **2.3**.

- Classe **[Préfixe]Data**

  Cette classe représente la description d'une instance de Vue Logique. Elle possède un ensemble de propriétés correspondant aux Rubriques de la Vue Logique.

- Classe **[Préfixe]SelectionCriteria**

  Cette classe représente la description des critères de sélection. Elle possède un ensemble de propriétés correspondant aux Rubriques de type identifiant et paramètres d'extraction associés à la Vue Logique.

- Classe **[Préfixe]Buffer**

  Cette classe représente la description des informations contextuelles. Elle possède un ensemble de propriétés correspondant aux Rubriques du buffer utilisateur.

- Classe **[Préfixe]DataUpdate**

  Cette classe hérite de la classe **[Préfixe]Data**. Par rapport à la classe parente, elle possède 2 propriétés supplémentaires dont les valeurs varient en fonction des modifications en cours. Ces deux propriétés sont les suivantes :

  - **action** : action de mise à jour qui peut valoir **Read**, **Modified**, **Created** ou **Deleted**. Cette propriété n'est visible que dans la liste de la propriété **UpdatedFolders**.
  - **updatedInstancesCount** : nombre de mises à jour serveur utiles associé à l'instance de Dossier concernée qui peut valoir de 1 à n.

- Classe **[Préfixe]UserData**

  Cette classe hérite de la classe **[Préfixe]Data**. Elle contient une propriété supplémentaire qui correspond à la Rubrique clé de l'instance parente.

- Classe **[Préfixe]TableModel**

  Cette classe n'est générée que si vous avez coché l'option **Utiliser Swing** lors de la génération. Elle hérite de la classe **Pacbase TableModel** et implémente le composant swing **TableModel**, servant à alimenter un tableau swing. Elle permet d'afficher la liste des instances de la classe **[Préfixe]Data** générées dans le tableau swing.

- Classe **[Préfixe]UpdateTableModel**

Cette classe n'est générée que si vous avez coché l'option **Utiliser Swing** lors de la génération. Elle hérite de la classe **Pacbase UpdateTableModel** et implémente le composant swing **TableModel**, servant à alimenter un tableau swing. Elle permet d'afficher la liste des instances de la classe **[Préfixe]DataUpdate** générées dans le tableau swing.

2.5.2.1. Documentation en ligne des classes générées

La documentation de l'interface publique des classes générées est directement intégrée au code sous forme de commentaires.

Vous pouvez donc consulter cette documentation en accédant directement au source de l'élément de l'interface publique souhaité dans le Workbench ou un browser VisualAge.

Vous pouvez également choisir de générer cette documentation à l'aide de l'utilitaire Javadoc livré avec le JDK.

2.5.2.1.1. Génération de la documentation

Il est possible de générer la documentation associée à un projet, un package, une classe ou encore une méthode.

Pour lancer Javadoc, vous disposez de deux solutions.

- Dans VisualAge, depuis le Workbench ou un browser :
  - sélectionnez le projet, le package, la classe ou la méthode souhaité(e).
  - accédez au menu contextuel associé et sélectionnez le choix **Generate javadoc**. Par défaut, la documentation est générée dans le répertoire de VisualAge, dans le sous-répertoire **…\Ide\javadoc**.
- A partir d'une session DOS ou OS/2, en utilisant les paramètres par défaut du générateur des composants Proxy, lancez la commande :

```
javadoc -classpath c:\vap\generated\java -d c:\doc -public
com.ibm.vap.generated.data com.ibm.vap.generated.proxies
```

2.5.2.1.2. Résultat obtenu

La documentation générée ne contient que des informations pertinentes pour le développeur car elle correspond exactement à chaque classe générée.

En effet, seule la documentation des éléments de l'interface publique - propriétés ou méthodes – **effectivement générés** – est extraite. Cette documentation est au format HTML et inclut les liens hypertextes.

Pour chaque méthode, outre les commentaires, sa signature – paramètre(s), code retour et exceptions - est également extraite.

A titre indicatif en voici un exemple :

☞ Vous pouvez également consulter le *Manuel de Référence Clients Graphiques : Interface Publique des composants générés* où les propriétés, méthodes et événements sont documentés par ordre thématique.

2.5.2.2. Personnalisation des classes

Toute classe générée hérite d'une classe générique.

Les classes génériques sont chargées une seule fois à l'installation du produit.

Si vous souhaitez implémenter de nouvelles fonctionnalités dans les composants Proxy générés, vous devez vous appuyer sur ce mécanisme d'héritage pour créer de nouvelles classes qui seront réservées à ces traitements. Les classes mères et les classes générées ne doivent pas être modifiées.

2.5.2.2.1. Personnalisation des classes ProxyLv

Il est possible d'ajouter un comportement commun à toutes les Proxy Racines, Références ou Dépendantes.

Il suffit pour cela de créer une classe héritant de **`DependentProxyLv`**, **`FolderProxyLv`** ou **`ReferenceProxyLv`**, d'y implémenter la nouvelle fonctionnalité souhaitée et de modifier ensuite la classe mère des objets Proxy générés.

☞ Cette dernière modification devra être refaite à chaque nouvelle génération.

2.5.2.2.2. Personnalisation des classes data

Il est possible de changer la classe **`Data`** utilisée par une Proxy donnée.

Pour cela, il faut définir une classe héritant de la classe **`DataDescription`**, y effectuer les implémentations nécessaires, puis modifier les méthodes **`newData()`** et **`newData(String[] values)`** de la Proxy.

3. Génération VisualAge Smalltalk

Pour développer un Client Pacbench C/S dans VisualAge, vous utilisez des classes générées mais également un grand nombre de classes génériques qui sont livrées avec le produit pour éviter la multiplication des éléments à chaque nouvelle génération. Contrairement aux classes générées documentées plus loin, les classes génériques ne dépendent pas des caractéristiques de la Vue Logique traitée.

☞ A l'installation, toutes ces classes ne sont pas visibles dans l'Organizer VisualAge. Pour atteindre votre point d'insertion, vous devez d'abord les afficher. Pour cela, dans le menu **`Applications`**, sélectionnez **`View`**, puis **`Show All Applications`**.

3.1. Classes génériques livrées

Vous devez vous assurer que les classes génériques sont installées sur votre station avant de commencer tout développement.

Les classes génériques à partir de la version 2.5 V08 (de type **`Vpcs`**) ne peuvent pas être utilisées avec des classes générées avec une version antérieure. Si vous voulez utiliser ces nouvelles classes génériques, il faudra impérativement générer à nouveau la totalité des classes de votre application afin qu'elle puisse être exécutée.

3.1.1. Communication

La communication de la partie cliente s'appuie sur une interface du Middleware de Pacbench C/S. Il est de plus possible de préciser différentes locations pour les cibles visées.

Ces classes sont indispensables à toute utilisation du produit que ce soit en exécution, édition ou génération.

3.1.1.1. Interface middleware

L'interface middleware permet de dialoguer avec les DLLs fournies avec le produit afin d'envoyer des messages au serveur ou d'en recevoir.

Les classes qui implémentent cette interface se trouvent dans l'application **`VapCommApi`**.

3.1.1.2. Gestionnaire de *locations*

Le gestionnaire de *locations* (ou localisations) permet de gérer les paramètres des différentes configurations serveur possibles.

Les classes qui implémentent le gestionnaire de *locations* se trouvent dans l'application **`VpcsLocationsApp`**.

3.1.2. Runtime

Les classes génériques du runtime sont nécessaires lors de l'exécution de toute application utilisant un Client Pacbench C/S. Ces classes sont réparties dans diverses applications ou sous-applications dépendant de la fonctionnalité implémentée.

3.1.2.1. Cache local

Le cache local permet de stocker et de gérer les instances de la partie cliente provenant du serveur ou devant l'alimenter.

Les classes qui implémentent le cache local se trouvent dans la sous-application **`VapCacheManagerSubApp`** de l'application **`VapRunGenericPartsApp`**.

3.1.2.2. Gestionnaire d'échange

Le gestionnaire d'échange permet de gérer les requêtes entre la partie cliente et le middleware.

Les classes qui implémentent le gestionnaire d'échange se trouvent dans la sous-application **`VapExchangeManagerSubApp`** de l'application **`VapRunGenericPartsApp`**.

3.1.2.3. Gestionnaire de communication

Le gestionnaire de communication permet de faciliter l'interface entre le gestionnaire d'échange et le middleware.

Les classes qui implémentent le gestionnaire de communication se trouvent dans la sous-application **`VapCommunication`** de l'application **`VapRunGenericPartsApp`**.

☞ La classe **`VapServerManagement`** représente le point d'entrée dans les traitements de communication. Elle possède une classe fille **`VapLUWServerManagement`** qui permet de gérer la communication.

Cette classe est spécialisée par **`VapLUWServerUserManagement`** qui permet à l'utilisateur d'implémenter son propre protocole de communication.

☞ Pour des informations sur la personnalisation du middleware, reportez-vous au chapitre 10, sous-chapitre ***VisualAge Smalltalk***.

3.1.2.4. Modèle des vues de dossier

Le modèle des Vues de Dossier définit un ensemble d'objets permettant de récupérer et de travailler sur les informations provenant du serveur.

Les classes qui implémentent ce modèle se trouvent dans les applications **`VpcsCommonRuntimeApp`** et **`VpcsRuntimeApp`** et leurs sous-applications.

☞ Pour conserver la compatibilité avec les versions antérieures à la version 2.5 V08, les anciennes classes du modèle ont été conservées dans la sous-application **`VapGenericProxysSubApp`** de l'application **`VapRunGenericPartsApp`**.

Parmi ces classes, la classe **`VapUserServiceError`** liée à la gestion des erreurs est personnalisable.

3.1.3. Edition

Les classes d'édition sont les classes qui permettent de gérer et manipuler l'interface graphique. Elles ne sont utilisées qu'à l'édition des objets correspondants dans le Composition Editor.

On suppose, dans cette phase d'édition, que la phase de génération a déjà été exécutée mais il n'est pas obligatoire d'avoir chargé le générateur pour l'utiliser.

Les classes d'édition se trouvent dans les applications **VpcsCommonEditionApp** et **VpcsEditionApp** et leurs sous-applications.

☞ Pour conserver la compatibilité avec les versions antérieures à la version 2.5 V08, les anciennes classes de l'édition ont été conservées dans l'application **VapEditGenericPartsApp** et ses sous-applications.

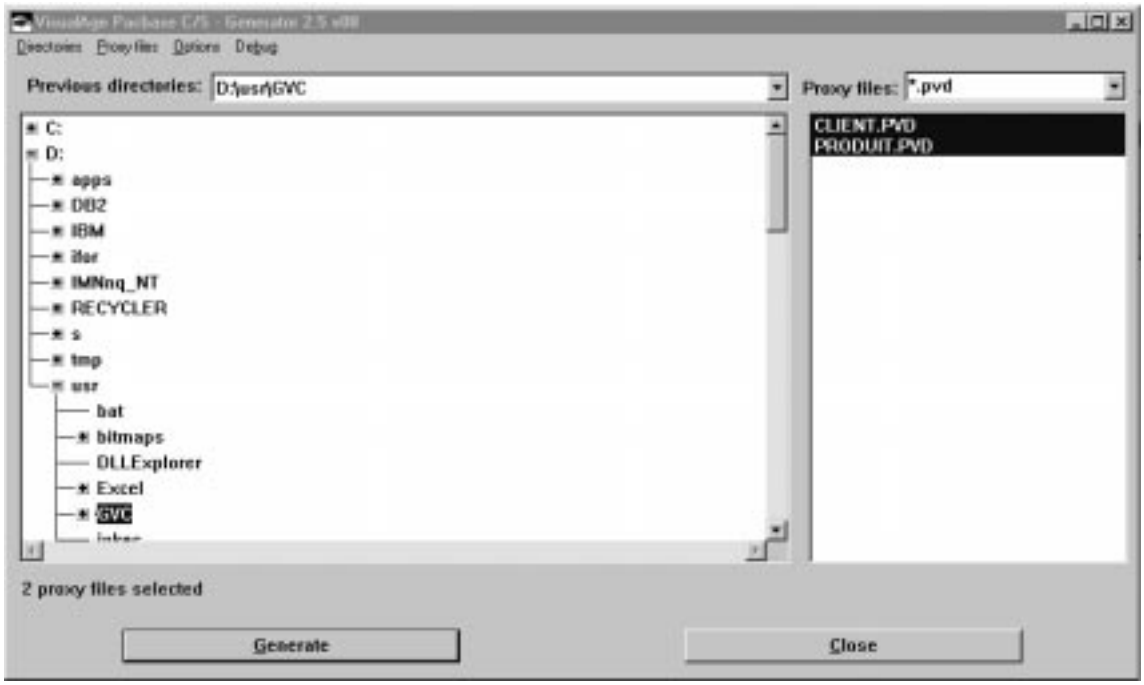
3.2. Fenêtre de génération

3.2.1. Lancement

Le générateur de Pacbench C/S permet de générer des Proxy Vues de Dossier.

Le lancement de la génération se fait à partir de la fenêtre **Générateur** de Pacbench Client/Serveur accessible via le choix **Pacbase C/S Generator** du menu **Tools** dans l'Organizer de VisualAge.

La fenêtre principale s'ouvre :



3.2.2. Sélection des fichiers

- Il faut tout d'abord renseigner le chemin d'accès au(x) fichier(s) à générer :
  - en les sélectionnant dans la liste déroulante de mémorisation des répertoires **Previous directories**.

    Cette liste s'enrichit implicitement lorsque vous appuyez sur le bouton **Generate**. Elle peut aussi être mise à jour explicitement à l'aide du menu **Directories** ou du menu contextuel.
  - soit en les sélectionnant dans l'arborescence des répertoires.

- Sélectionnez ensuite le ou les fichiers à générer dans la liste à choix multiples. Le nombre de fichiers valides sélectionnés est directement visualisable dans la ligne d'information en bas de la fenêtre.

☞ Les fichiers affichés dans la liste peuvent être filtrés en sélectionnant un nom de fichier générique dans la combo-box **Proxy files**. L'utilisateur peut enrichir cette liste avec des filtres personnalisés qui seront mémorisés lors de la génération. Elle peut aussi être réduite à l'aide du menu **Proxy files** ou du menu contextuel.

3.2.3. Description des menus

La fenêtre contient trois menus principaux.

3.2.3.1. Le menu Directories

Ce menu comprend les choix suivants :

- **Add Selected Directory To Recall List** : sauvegarde le répertoire sélectionné dans la liste déroulante de mémorisation.
- **Remove Directory From Recall List** : supprime le répertoire affiché de la liste déroulante de mémorisation.

3.2.3.2. Le menu Proxy files

- **Generate** : génère le ou les fichiers sélectionnés. Ce choix n'est accessible que si au moins un des fichiers sélectionnés est valide.
- **View Log** : permet de visualiser le dernier compte-rendu de génération du ou des fichiers sélectionnés. Ce choix n'est actif que si au moins un des fichiers sélectionnés possède un fichier compte-rendu. Les comptes-rendus de générations sont stockés dans le même répertoire que les fichiers à générer, ont le même nom que le fichier à générer mais sont caractérisés par l'extension **.log**.
- **Remove Proxy File Wildcard :** permet de supprimer un nom de fichier générique de la combo-box **Proxy Files**.

☞ Les choix de ce menu sont accessibles également par le menu contextuel de la liste des fichiers à générer.

3.2.3.3. Le menu Options

Ce menu comprend les options générales suivantes :

- **View Log After Generation** : permet d'afficher ou non le compte-rendu directement après la génération (**activé par défaut**)
- **Force Library Update** : permet de forcer la mise à jour des propriétés dans la librairie Smalltalk même si aucune différence n'est constatée (**désactivé par défaut**)
- **Generate UserContext Classes** : permet de générer les classes **UserContext** (**activé par défaut**).
- **Optimize Generated Code** : permet d'optimiser la génération des propriétés des classes de type Proxy (**désactivé par défaut**). Quand cette option est utilisée, les propriétés des composants Proxy sont réduites au minimum fonctionnel lors de la génération.

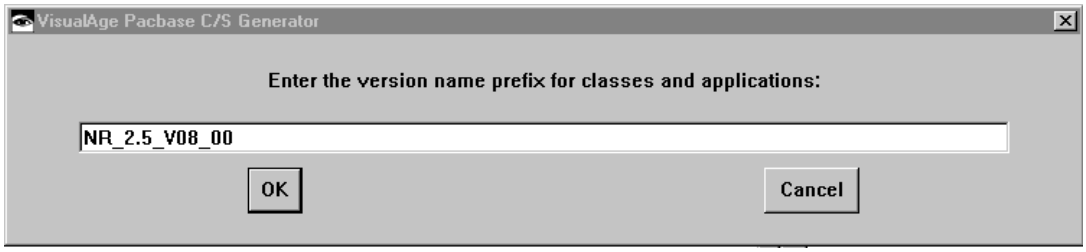
  

  Lorsque cette option est activée, certaines propriétés optionnelles ne sont pas générées.

- **Bypass Repository Verification** : permet d'inhiber le contrôle concernant l'origine (base et bibliothèque VisualAge Pacbase) du fichier à générer (**désactivé par défaut**). Si cette option est positionnée et que le contrôle échoue, la génération ne sera pas interrompue, seul un message d'avertissement sera ajouté dans le compte-rendu.

Le menu **Options** comprend également des options de gestion des applications lors de la génération.

- **Set Version Name** : permet d'ouvrir une boîte de dialogue dans laquelle on peut spécifier le code de la version qui sera utilisée pour versionner les classes et les applications :



Les trois choix suivants permettent à l'utilisateur de préciser le comportement du générateur vis à vis des applications ou sous-applications impliquées lors de la génération.

- **Application Edition Creation Mode**

  Ce choix comprend un sous-menu avec les choix suivants :
  - **Always Create New Edition** : si ce choix est activé, les applications versionnées passeront automatiquement en édition.
  - **Confirm Before Creating Edition** : les applications versionnées passeront en édition après confirmation de l'utilisateur (**mode par défaut**).
  - **Confirm Before Creating Scratch Edition** : les applications versionnées passeront en scratch edition après confirmation de l'utilisateur.
  - **Stop Generation If Not an Edition** : Arrêt immédiat de la génération à la première application rencontrée qui n'est pas en mode édition.
- **Application Edition Versionning Mode**

  Ce choix comprend un sous-menu avec les choix suivants :
  - **Always Version** : Les applications en édition seront automatiquement versionnées.
  - **Confirm Before Versionning** : Les applications en édition seront versionnées après confirmation de l'utilisateur.
  - **Never Version** : Les applications ne seront pas versionnées (**mode par défaut**).

    ☞ Ces choix ne sont pas accessibles si l'utilisateur a choisi le mode scratch pour ses applications.
- **Application Version Release Mode**

  Ce choix comprend un sous-menu avec les choix suivants :
  - **Always Release** : Les applications versionnées constitueront automatiquement une release.

- **`Confirm Before Releasing`** : Les applications versionnées passeront automatiquement en release après confirmation de l'utilisateur.
- **`Never Version`** : Les applications ne passeront pas en release (**mode par défaut**).
  - ☞ Ces choix ne sont pas accessibles si l'utilisateur a choisi le mode scratch pour ses applications.

Les valeurs de toutes ces options sont automatiquement mémorisées et seront les valeurs courantes à la prochaine ouverture de la fenêtre de génération. Les valeurs par défaut indiquées sont donc des valeurs initiales.

3.2.4. Déroulement d'une génération

La barre de messages située au-dessus des deux boutons affiche des messages d'information sur le déroulement de la génération ou des messages d'erreur, le cas échéant.

Une fois les différents paramètres et options positionnés dans la fenêtre principale, déclenchez la génération.en cliquant sur le bouton **`Generate`**.

Le bouton **`Generate`** est inactif si aucun des fichiers sélectionnés n'est valide.

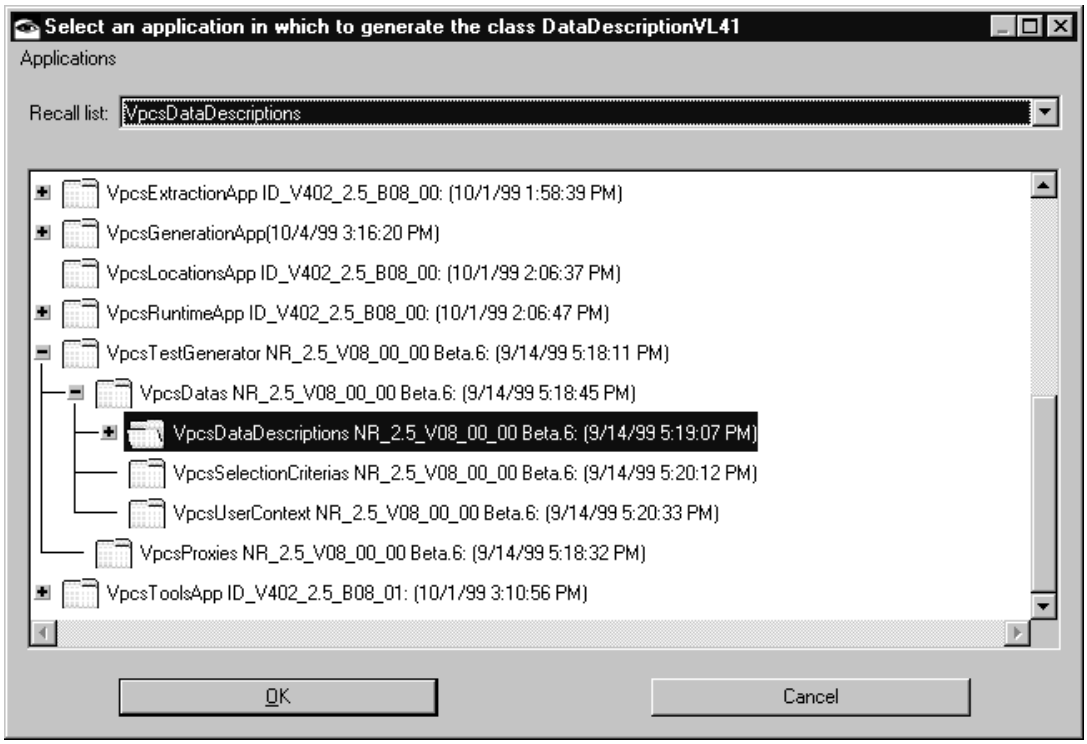
Si au moins une des classes à générer n'est pas présente dans l'image, alors une fenêtre de sélection s'ouvre afin d'affecter ces nouvelles classes à une application.

Voir ci-dessous le paragraphe *Affectation des applications*.

3.2.5. Affectation des applications

Cette fenêtre s'affiche si au moins une des classes à générer n'existe pas dans l'image VisualAge.

Toutes les classes **`ProxyLv`** d'un même Dossier seront générées dans la même application que la Proxy Racine ; en revanche, les classes **`data`** appartenant à un même Dossier peuvent être affectées à des applications différentes.

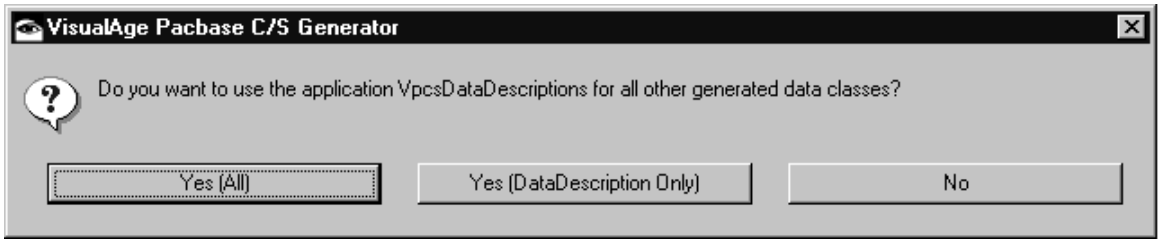


Cette fenêtre permet de spécifier dans quelle application VisualAge la classe dont le nom est rappelé dans la barre de titre va être générée.

- Sélectionnez l'application :
  - en la choisissant dans la liste des applications ou dans la liste de mémorisation,
  - ou en la créant de la façon suivante :
    - Dans le menu contextuel ou dans le menu **`Applications`**, sélectionnez le choix **`New…`**.
    - saisissez le nom de la nouvelle application et validez.

- Validez la sélection en cliquant sur OK.

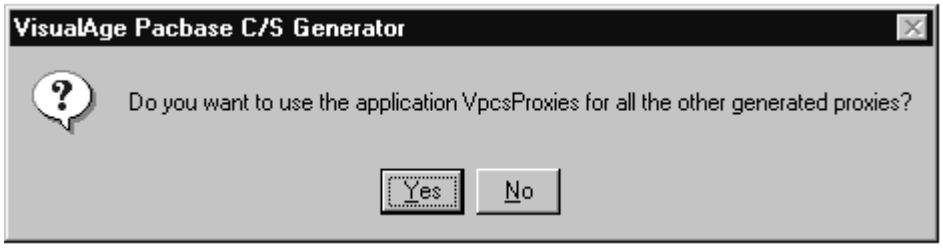
  Une boîte de dialogue s'affiche alors :
  - Pour les classes de type **`data`** :



- **`Yes (All)`** : Toutes les classes **`data`** non encore rattachées à une application seront générées dans l'application précédemment sélectionnée ou créée.
- **`Yes (type Only)`** : Toutes les classes **`data`** **de même type** non encore rattachées à une application seront générées dans cette application.

- **No** : L'application sélectionnée ne sera pas affectée aux autres classes **data** à générer.

- Pour les classes **ProxyLv** :



- **Yes** : Toutes les classes **ProxyLv** non encore rattachées à une application seront générées dans l'application sélectionnée ou créée.
- **No** : L'application sélectionnée ne sera pas affectée aux autres classes **ProxyLv** à générer.

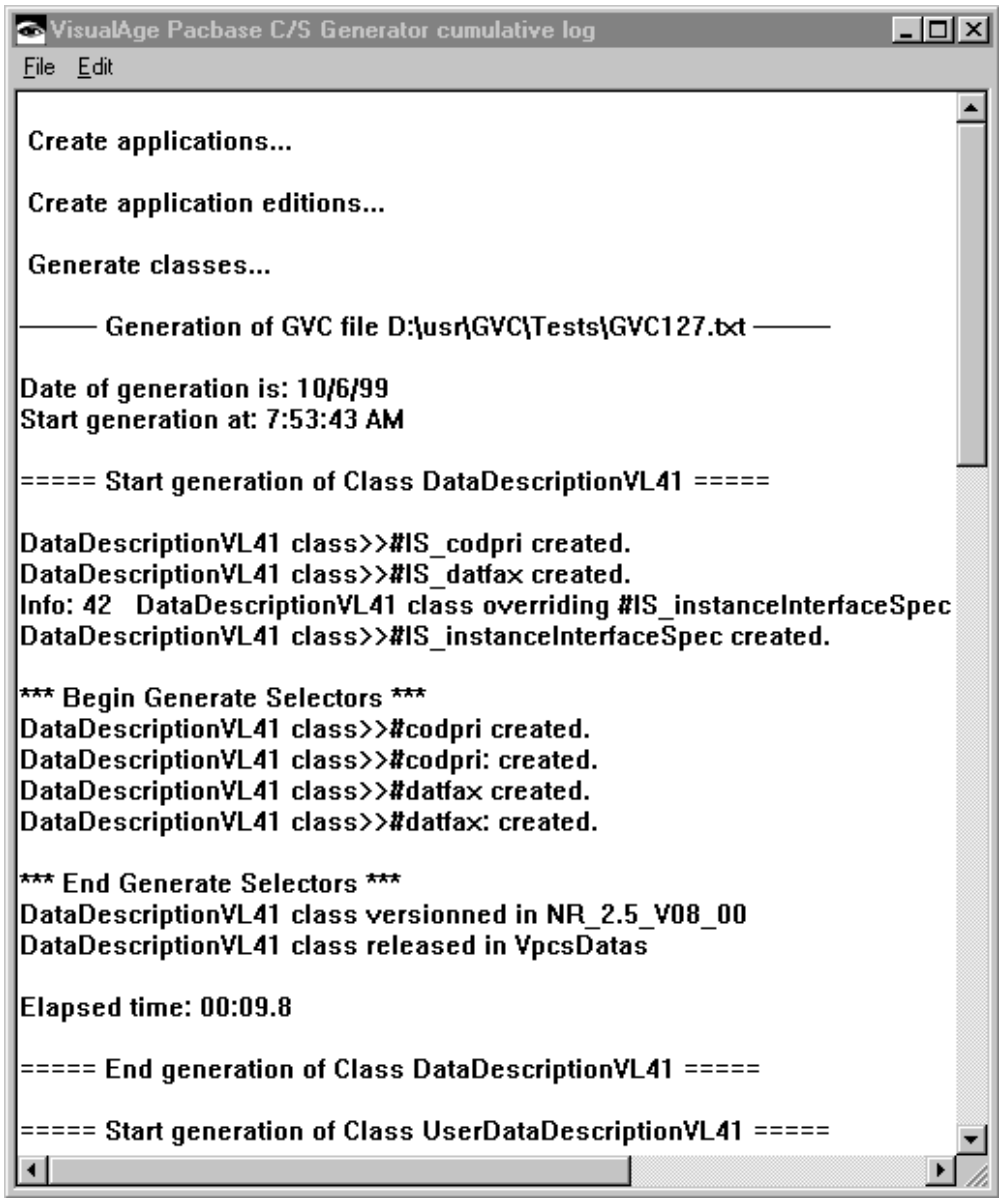
☞ L'entité utilisée pour l'affectation d'application étant la Proxy Vue de Dossier, toutes les classes **ProxyLv** composant cette Proxy Vue de Dossier sont automatiquement affectées à cette même application.

Rappel Les classes **ProxyLv** générées correspondent aux Proxy Vues de Dossier ou Proxy Elémentaires. Les classes **data** ou classes de données désignent les classes générées correspondant aux différents types de données de la Vue Logique : **DataDescription**, **SelectionCriteria**, **UserContext**, **UserDataDescription**, **UpdatedDataDescription** **ValuesOf** et **ValuesOf(...)Converter**.

3.2.6. Compte-rendu de génération

Le compte-rendu issu de la génération est affiché automatiquement dans un workspace dès lors que l'option **View Log After Generation** est activée. Il est possible, dans le cas contraire, de visualiser le ou les rapports de génération par le choix **View Log.**

Le compte-rendu décrit les différentes phases de la génération.



Dans la phase de génération des classes, le rapport présente pour chaque fichier généré :

- Le nom complet de celui-ci :

---------- Generation of GVC file D:\usr\GVC\Tests\GVC127.txt ----------

- La date et l'heure de début de génération

Date of generation is: 10/6/99

Start generation at: 07:53:43 AM

- Le descriptif de génération de chaque classe :
  - Le nom des méthodes créées, remplacées, supprimées, surchargées ou inchangées :

DataDescriptionVL41 class>>#IS_codpri created.

  - le code de la version, et le nom de l'application dans laquelle elle est releasée, le cas échéant :

DataDescriptionVL41 class versionned in NR_2.5_V08_00_00a

DataDescriptionVL41 class released in VpcsDatas

- la durée de génération de la classe
- Le message de fin de génération de la classe :

===== End generation of Class DataDescriptionVL41 =====

La codification et le contenu des éléments générés sont documentés dans le sous-chapitre 3.3, *Résultats de la génération*.

3.3. Résultats de la génération

3.3.1. Introduction

Les éléments générés dépendent toujours du type de service effectué par le Composant Applicatif. Ils ne seront pas les mêmes selon que le Composant Applicatif effectue des services de mise à jour ou simplement de lecture.

3.3.2. Classes générées

3.3.2.1. Codification des classes générées

Dans VisualAge, les éléments générés par Pacbench Client/Serveur correspondent à des classes dont la codification dépend de paramètres positionnés dans le Référentiel. Dans la suite du chapitre, ces paramètres sont représentés par les éléments suivants :

- **`Préfixe`** — Code de la Vue de Dossier (8 car. max.) ou du Composant Applicatif dans un développement mono-vue (si l'option **`PREFIX`** n'est pas renseignée)
- **`CodeClasse`** — Code de classe des Proxy Elémentaires spécifiée par le paramètre **`CLASSCODE`** (20 car. max.) de l'écran Commentaires du Dossier ou du Composant Applicatif dans un développement mono-vue
- **`CodeVueLogique`** — Code de la Vue Logique (4 car.)
- **`CodeBufferUtilisateur`** — Code du buffer utilisateur (4 car.)
- **`Suffixe`** — Suffixe de Vue Logique ou buffer utilisateur spécifié par :
  - le paramètre **`SUFF`** (20 car. max.) dans l'écran Commentaires de la Vue Logique ou du Segment buffer utilisateur pour les classes **`DataDescription`**, **`SelectionCriteria`**, **`UserContext`**, **`UpdatedDataDescription`**, **`UserDataDescription`**, **`ValuesOf`** et **`ValuesOf(...)Converter`**.
  - le paramètre **`PROXYSUF`** (20 car. max.) dans l'écran Commentaires du

Composant Applicatif (valeur par défaut **ProxyLv**).

- **CodeRubrique** Code de la Rubrique

3.3.2.2. Contenu des classes générées

Les paramètres entre **[...]** sont variables.

- classe **[Préfixe][CodeClasse][ProxyLv]**

Cette classe représente une Proxy Elémentaire Racine, Dépendante ou Référence.

☞ Dans un développement mono-vue, il s'agit obligatoirement d'une Proxy Racine.

- classe **DataDescription[CodeVueLogique][Suffixe]**

Cette classe représente la description d'une instance de Vue Logique. Elle possède un ensemble d'attributs correspondant aux Rubriques de la Vue Logique.

- classe **SelectionCriteria[CodeVueLogique][Suffixe]**

Cette classe représente la description des critères de sélection. Elle possède un ensemble d'attributs correspondant aux Rubriques de type identifiant et paramètres d'extraction associés à la Vue Logique.

- classe **UserContext[CodeBufferUtilisateur][Suffixe]**

Cette classe représente la description des informations contextuelles. Elle possède un ensemble d'attributs correspondant aux Rubriques du buffer utilisateur.

- classe **UpdatedDataDescription[CodeVueLogique][Suffixe]**

Cette classe hérite de la classe **DataDescription[CodeVueLogique][Suffixe]**. Par rapport à la classe parente, elle possède 2 attributs supplémentaires dont les valeurs varient en fonction des modifications en cours. Ces deux attributs sont les suivants :

  - **action** : action de mise à jour qui peut valoir **Read**, **Modified**, **Created** ou **Deleted**. Cet attribut n'est visible que dans la liste de l'attribut **UpdatedFolders**.
  - **updatedInstancesCount** : nombre de mises à jour serveur utiles associé à l'instance de Dossier concernée qui peut valoir de 0 à n.

- classe **UserDataDescription[CodeVueLogique][Suffixe]**

Cette classe hérite de la classe **DataDescription[CodeVueLogique][Suffixe]**. Elle contient un attribut supplémentaire qui correspond à la Rubrique clé de l'instances parente.

- classe **ValuesOf[CodeRubrique][Suffixe]**

Cette classe est générée pour chaque Rubrique qui contient des tables de valeurs. Elle hérite de la classe générique **VpcsValues**.

- classe **`ValuesOf[CodeRubrique][Suffixe]Converter`**

Cette classe, associée à la précédente, est utilisée comme une aide à la saisie, en permettant la conversion et le formatage des valeurs que l'utilisateur entre. Elle hérite de la classe générique **`VpcsValuesConverter`**.

3.3.2.3. Personnalisation des classes

Toute classe générée hérite d'une classe générique.

Les classes génériques sont chargées une seule fois à l'installation du produit.

Si vous souhaitez implémenter de nouvelles fonctionnalités dans les composants Proxy générés, vous devez vous appuyer sur ce mécanisme d'héritage pour créer de nouvelles classes qui seront réservées à ces traitements. Les classes mères et les classes générées ne doivent pas être modifiées.

- Pour modifier le comportement d'une Proxy donnée :

  La Proxy ne doit pas être modifiée. Vous devez créer une nouvelle classe héritant de de la classe **`ProxyLv`** générée et redéfinir les méthodes de ladite classe.

- Pour modifier le comportement de plusieurs Proxy et leur définir un nouveau comportement qui leur soit commun :

  Vous devez créer une nouvelle classe par composant à modifier. Cette classe doit hériter de la classe **`ProxyLv`** générique correspondante :

  - **`VpcsFolder`**, pour une Proxy Racine,
  - **`VpcsDependentProxyLv`**, pour une Proxy Dépendante,
  - **`VpcsReferenceProxyLv`**, pour une Proxy Référence.

  Il suffit alors de surcharger les méthodes de chaque classe mère nouvellement créée.

Pour connaître le schéma d'héritage des classes **`ProxyLv`** générées, reportez-vous au *Manuel de Référence Clients Graphiques : Interface Publique des composants générés*.

4. Génération pour cibles COM

Une Proxy COM est un objet COM capable de relayer des services implémentés sur un serveur distant. Cet objet peut être appelé par des outils construisant des clients graphiques et intégrant des objets COM.

Avant de générer les objets Proxy, vous devez extraire du Référentiel les Composants Applicatifs associés. Pour plus d'informations à ce sujet, reportez-vous au *Guide de l'Utilisateur, Vol. II – Services Applicatifs.*

La génération de Proxy Vues Logiques n'est pas traitée dans ce manuel. Si vous souhaitez néanmoins des détails sur ce mode de génération, référez-vous à la version antérieure du *Guide de l'Utilisateur Pacbench Client/Serveur : Clients Graphiques.*

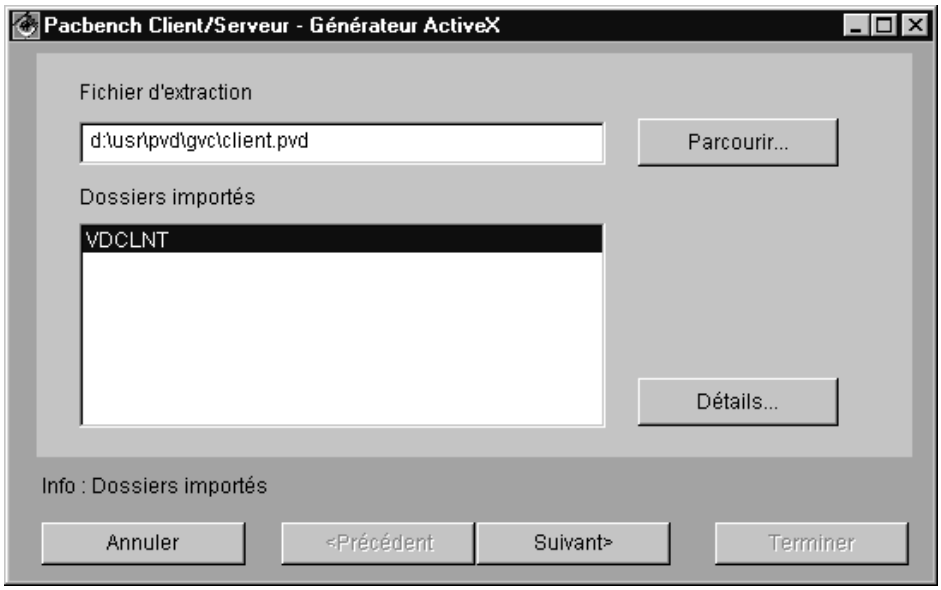
4.1. Lancement du générateur en mode graphique

Le générateur ActiveX fonctionne sous Windows/NT, Windows 95 et Windows 98. Il permet de générer des composants Proxy COM.

Pour activer le Générateur, vous devez lancer le programme **`olegen.exe`** dans le groupe de programmes spécifique créé lors de l'installation.

Sur la ligne de **`Command Line Argument`**, saisissez l'option langue souhaitée comme suit : **`-lang fr`** ou **`–lang en`**. Si la zone est à blanc, l'interface du générateur sera en français.

Le formulaire principal du générateur s'ouvre :



- **Sélection des Vues de Dossier à générer**

Vous devez d'abord renseigner le chemin d'accès au fichier d'extraction dans la zone **Fichier d'extraction**. Vous pouvez :

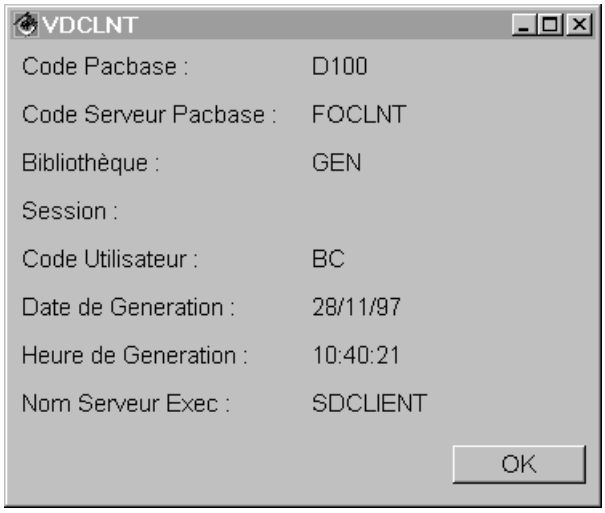
- le saisir dans la boîte edit, ou
- cliquer sur le bouton **Parcourir...**. Une fenêtre s'ouvre alors dans laquelle vous pouvez sélectionner le chemin complet du fichier à l'aide des différentes boîtes de liste.

Une fois le nom de fichier saisi, la liste de la ou des Vue(s) de Dossier contenue(s) dans le fichier d'extraction s'affiche dans la zone **Dossiers importés**.

Lorsqu'il existe plusieurs Vues de Dossier dans la liste, elles sont toutes automatiquement sélectionnées. Le cas échéant, désélectionnez celle(s) que vous ne souhaitez pas générer.

☞ Le message **Dossiers importés** indique que le fichier d'extraction est correct et que les Vues de Dossier qu'il contient sont identifiées et prêtes pour la génération.

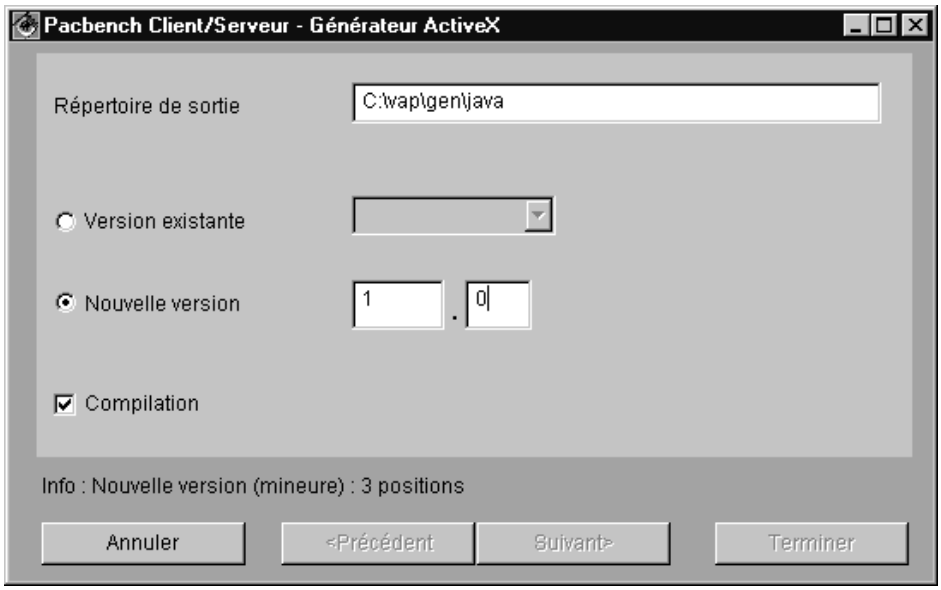
- **Autres fonctionnalités du formulaire**
  - Le bouton **Détails…** permet d'afficher le contexte de génération **GVC** pour la Vue de Dossier sélectionnée.



  - Le bouton **Suivant** provoque une navigation vers le formulaire qui vous permettra de spécifier les paramètres de génération. Pour des détails sur ces paramètres par défaut, reportez-vous au paragraphe suivant.
  - Si vous souhaitez générer immédiatement toutes les Vues de Dossier sélectionnées, en utilisant les paramètres par défaut, cliquez sur le bouton **Terminer**. Une boîte de message vous demande de confirmer la création du répertoire de sortie s'il n'existe pas encore. Cliquez sur OK ; la génération est lancée.

- **Options de génération**

Si vous avez cliqué sur **Suivant** dans le formulaire principal, le formulaire suivant s'affiche :



Ce formulaire vous permet de spécifier les paramètres de génération.

- Le champ **Répertoire de sortie** contient le chemin d'accès au répertoire dans lequel les classes seront générées.
- L'option **Version existante** permet de sélectionner, parmi les versions existantes, la version à générer.
- L'option **Nouvelle version** permet de spécifier une nouvelle version de génération des classes. La première zone de saisie comprend cinq chiffres maximum et la deuxième zone comprend trois chiffres maximum.
- L'option **Compilation** permet de lancer la compilation des sources C++ directement après leur génération.

Ce formulaire contient également les boutons-poussoirs suivants :

- Le bouton **Précédent** provoque la navigation vers le formulaire principal.
- Le bouton **Suivant** provoque la navigation vers le formulaire de modification des préfixes des classes générées.
- Si vous souhaitez générer immédiatement toutes les Vues de Dossier sélectionnées, en utilisant les noms de classes par défaut, cliquez sur le bouton **Terminer**. Une boîte de message vous demande de confirmer la création du répertoire de sortie s'il n'existe pas déjà. Cliquez sur OK ; la génération est lancée.

4.2. Lancement du générateur en mode batch

Pour lancer le générateur en mode batch, saisissez la commande :

olegenbatch –i<fichierextract> -o<répsortie> [**-v<version>**] [**-c**]

Le paramètre **-i** contient le chemin complet du fichier d'extraction

Le paramètre **-o** contient le répertoire de sortie des classes générées.

Le paramètre **-v** (optionnel) contient le numéro de version (divisé en deux parties séparées par un point. La première partie comprend 5 chiffres max. et la deuxième 3 chiffres max.) Par défaut, c'est la version serveur spécifiée dans VisualAge Pacbase qui sera prise en compte si elle existe. Sinon, c'est la dernière version générée qui sera prise en compte s'il en existe une. Si aucune de ces version n'existe, la valeur affectée par défaut est **1.0**.

Le paramètre **-c** (optionnel) indique une demande de compilation.

4.3. Résultats de la génération

Les éléments générés dépendent toujours du type de service effectué par le Composant Applicatif. Ils ne seront pas les mêmes selon que le Composant Applicatif effectue des services de mise à jour ou simplement de lecture.

Le fichier généré contient uniquement les classes qui dépendent des caractéristiques de la Vue Logique traitée.

4.3.1. Introduction

A l'issue de la génération, les fichiers suivants sont créés :

- Le fichier des localisations, **VAPLOCAT.INI** est créé dans le répertoire de sortie de génération.

Ce fichier doit être enrichi manuellement. Pour des informations sur la manière de procéder, reportez-vous à la section **7.3.3**.

Ce fichier doit être fourni en entrée de la gateway.

- Les fichiers sources des classes générées et ressources nécessaires à leur édition dans le répertoire **<Nomdossier><version>** (dans notre exemple : **VDCLNT2.0**).

4.3.2. Classes générées

Dans VisualAge, les éléments générés par Pacbench Client/Serveur correspondent à des classes dont la codification est composée d'un préfixe et d'une partie codée en dur dans le générateur. Le préfixe correspond au nom en clair de la Vue Logique qui comporte 36 caractères au maximum.

- Classe **[Préfixe]Data**

  Cette classe représente la description d'une instance de Vue Logique. Elle possède un ensemble d'attributs correspondant aux Rubriques de la Vue Logique.

- Classe **[Préfixe]SelectionCriteria**

  Cette classe représente la description des critères de sélection. Elle possède un ensemble d'attributs correspondant aux Rubriques de type identifiant et paramètres d'extraction associés à la Vue Logique.

- Classe **[Préfixe]Buffer**

  Cette classe représente la description des informations contextuelles. Elle possède un ensemble d'attributs correspondant aux Rubriques du buffer utilisateur.

- Classe **[Préfixe]DataUpdate**

Cette classe hérite de la classe **[Préfixe]Data**. Par rapport à la classe parente, elle possède 2 attributs supplémentaires dont les valeurs varient en fonction des modifications en cours. Ces deux attributs sont les suivants :

- **action** : action de mise à jour qui peut valoir **Read**, **Modified**, **Created** ou **Deleted**. Cet attribut n'est visible que dans la liste de l'attribut **UpdatedFolders**.
- **updatedInstancesCount** : nombre de mises à jour serveur utiles associé à l'instance de Dossier concernée qui peut valoir de 1 à n.

- Classe **[Préfixe]UserData**

  Cette classe hérite de la classe **[Préfixe]Data**. Elle contient un attribut supplémentaire qui correspond à la Rubrique clé de l'instances parente.

- Classe **[Préfixe]VapError**

  Cette classe regroupe les informations sur les problèmes divers pouvant intervenir en cours d'exécution de programme : clé d'erreur, libellé d'erreur et gravité.

Pour plus d'informations, consultez le *Manuel de Référence Clients Graphiques : Interface Publique des Composants Générés.*

Une description sommaire de l'interface publique est disponible en ligne. Vous pouvez la visualiser si votre outil client le permet.

4.4. Résultats de la compilation

Si la compilation s'est déroulée correctement, les éléments compilés se trouvent dans le répertoire **<Nomdossier><version>\Release** (dans notre exemple **VDCLNT2.0\Release**).

Un compte-rendu est édité dans le fichier **<Nomdossier><version>\Resume.txt**.

Vous pouvez également exécuter la compilation à partir de Visual C++. Pour connaître la marche à suivre, consultez le sous-chapitre **4.5** *Compilation avec la version 5.0 et 6.0 de Visual C++*

Une fois la génération et la compilation effectuées, la proxy est directement exploitable sur la machine où est installé le générateur.

4.5. Compilation avec la version 5.0 et 6.0 de Visual C++

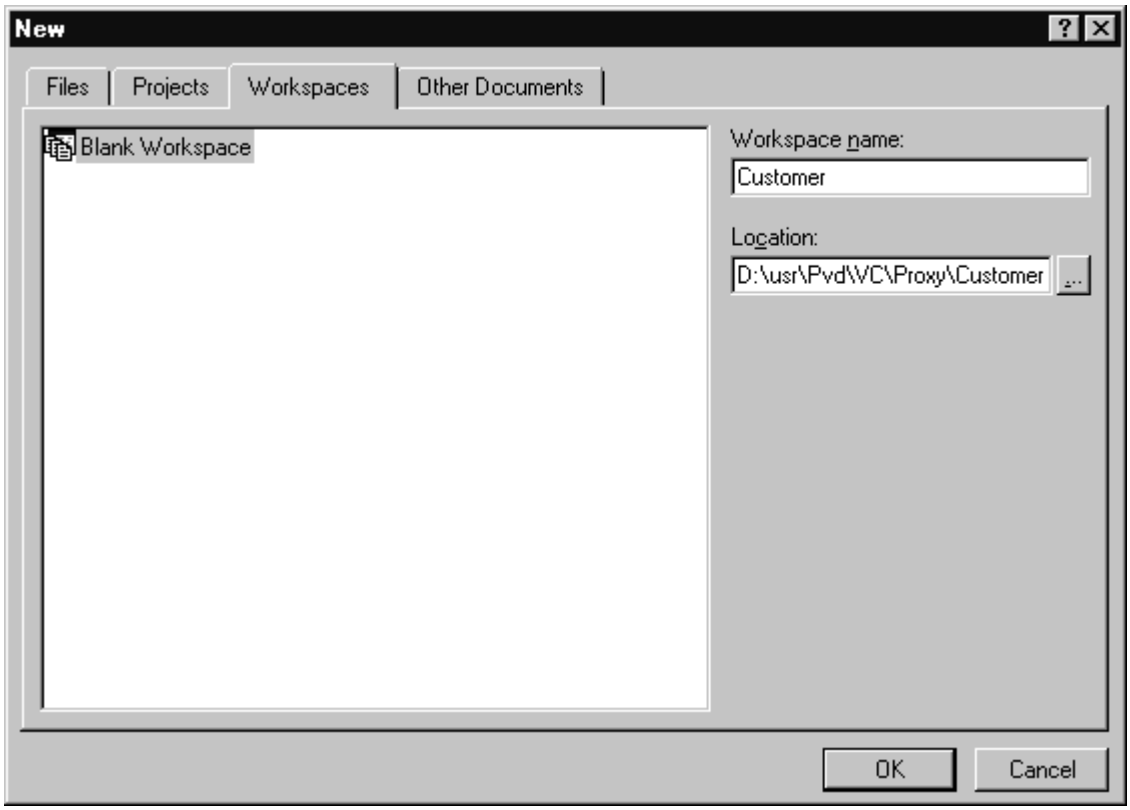
Suite à la génération, vous obtenez un ensemble de sources que vous allez utiliser pour compiler votre Proxy.

A partir de Visual Studio C++, vous devez créer dans un premier temps un nouveau Workspace qui contiendra deux projets, le premier correspondant à la librairie statique de la proxy et le deuxième à la proxy elle même. Pour ce faire effectuez les étapes suivantes.

4.5.1. Création d'un nouveau workspace

- Cliquez sur le menu **File** de la barre d'outils et sélectionnez le sous-menu **New** dans le menu déroulant.

- Ouvrez l'onglet **Workspace** dans la boîte de dialogue **New** et sélectionnez **Blank Workspace** afin de créer un projet vierge. Dans le champ **Workspace Name**, entrez le nom souhaité pour votre projet et dans le champ **Location**, le nom du répertoire dans lequel vous désirez le créer, puis cliquez sur **OK**.



A l'issue de cette étape, un sous-répertoire est créé sous le répertoire que vous avez sélectionné dans le champ **Location**. Le nom de ce sous-répertoire correspond au nom du projet que vous avez précédemment saisi dans le champ **Workspace name**.

4.5.2. Création du projet de la librairie statique

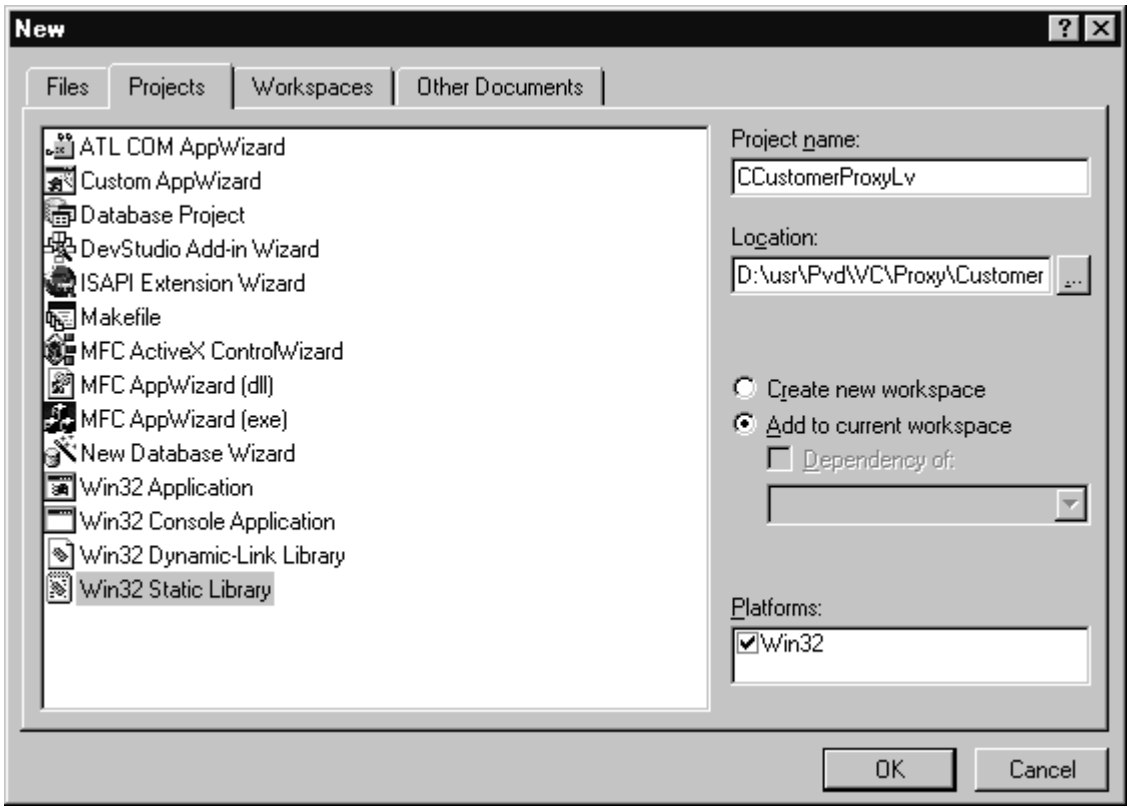
- Cliquez sur le menu **File** de la barre d'outils et sélectionnez le sous-menu **New** dans le menu déroulant.

- Cliquez sur l'onglet **Projects** dans la boîte de dialogue **New** et sélectionnez **Win32 Static Library** afin d'ajouter au Workspace précédemment créé un nouveau projet qui correspondra à la librairie statique utilisée par la proxy.

- Dans **Project Name,** entrez le nom du projet en respectant la syntaxe suivante **C<nomNoeudRacine>ProxyLv**.

- Dans **Location**, sélectionnez le sous-répertoire qui a été automatiquement créé lors de la création du workspace vierge.

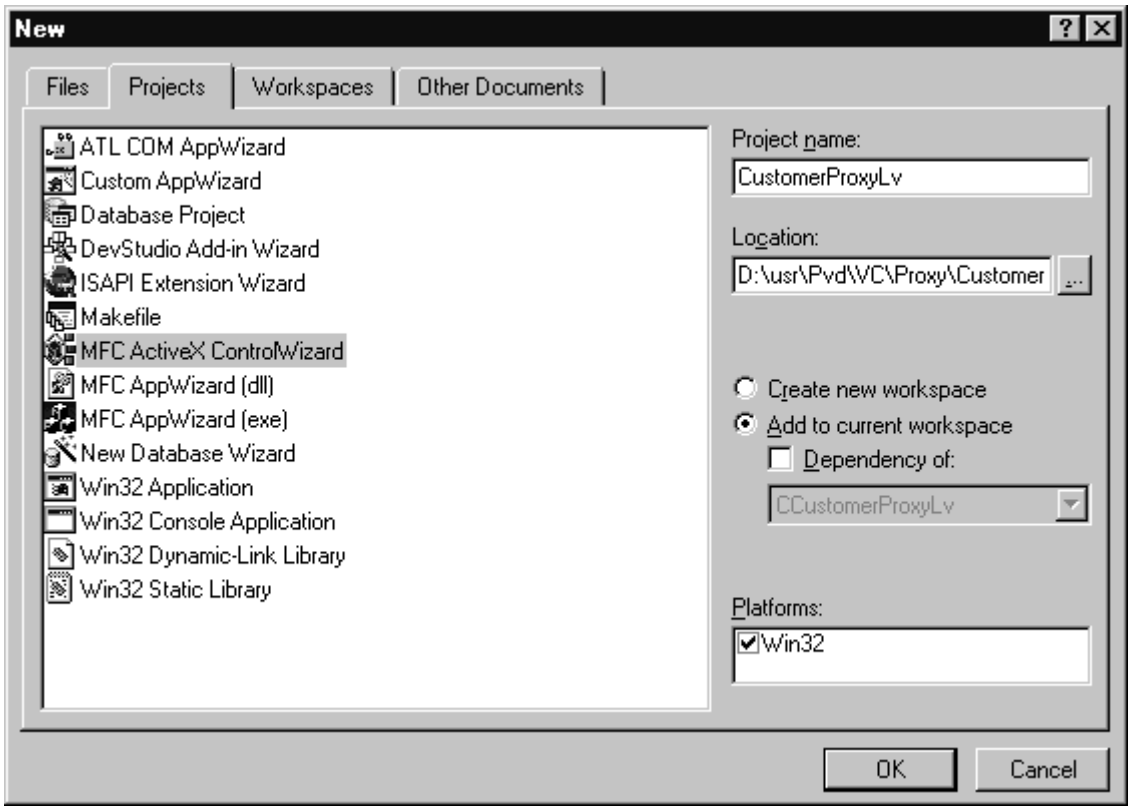
- Sélectionnez le bouton radio **Add to current workspace** puis cliquez sur **OK**.



☞ En Visual C++ 6.0, une fenêtre s'affiche, proposant des options. Ne sélectionnez rien, cliquez sur **`Finish`** puis **`OK`**.

4.5.3. Création du projet Proxy COM

- Cliquez sur le menu **`File`** de la barre d'outils et sélectionnez le sous-menu **`New`** dans le menu déroulant.
- Ouvrez l'onglet **`Projects`** dans la boîte de dialogue **`New`** et sélectionnez **`MFC ActiveX ControlWizard`** afin d'ajouter au Workspace précédemment créé un nouveau projet qui correspondra à la Proxy.
- Dans **`Project Name`**, entrez le nom du projet en respectant la syntaxe suivante **`<nomNoeudRacine>ProxyLv`**.
- Dans **`Location`**, sélectionnez le sous-répertoire qui a été automatiquement créé lors de la création du workspace vierge.
- Cochez le radio bouton **`Add to current workspace`** puis cliquez sur **`OK`**.
- Une nouvelle fenêtre apparaît. Cliquez sur **`Finish`** puis sur **`OK`**.



4.5.4. Transferts des fichiers

A l'issue des trois étapes précédentes, vous diposez des répertoires ci-dessous. Afin de rendre la suite de la procédure plus claire, nous attribuons un nom à chacun de ces répertoires, comme suit :

- *Folder1* : …\<nomWorkspace>
- *Folder2* : …\<nomWokspace>\C<nomNoeudRacine>ProxyLv
- *Folder3* : …\<nomWokspace>\<nomNoeudRacine>ProxyLv

Afin d'alimenter respectivement chacun des deux projets créés, vous devez copier certains des fichiers qui se trouvent dans le répertoire de génération de la Proxy à l'aide d'un gestionnaire de fichiers :

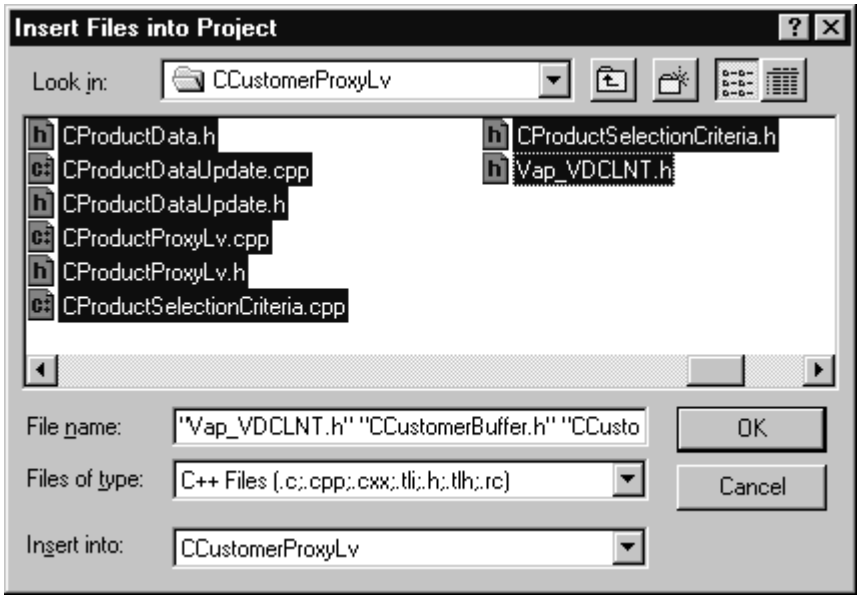
- Sélectionnez tous les fichiers préfixés par **`C<nomNoeud>…`** ainsi que le fichier **`VAP_<nomDossier>.h`**
- Copiez les fichiers sélectionnés dans le répertoire *Folder 2*.
- Copiez tous les autres fichiers dans le répertoire *Folder3*.

4.5.5. Ajout des fichiers aux projets

Vous devez maintenant ajouter les fichiers que vous venez de copier au projet de la librairie statique.

- Cliquez sur le menu **`Project`** de la barre d'outils et sélectionnez le sous-menu **`Set Active Project`** dans le menu déroulant.
- Sélectionnez le projet de la librairie statique **`C<nomNoeudRacine>ProxyLv`**

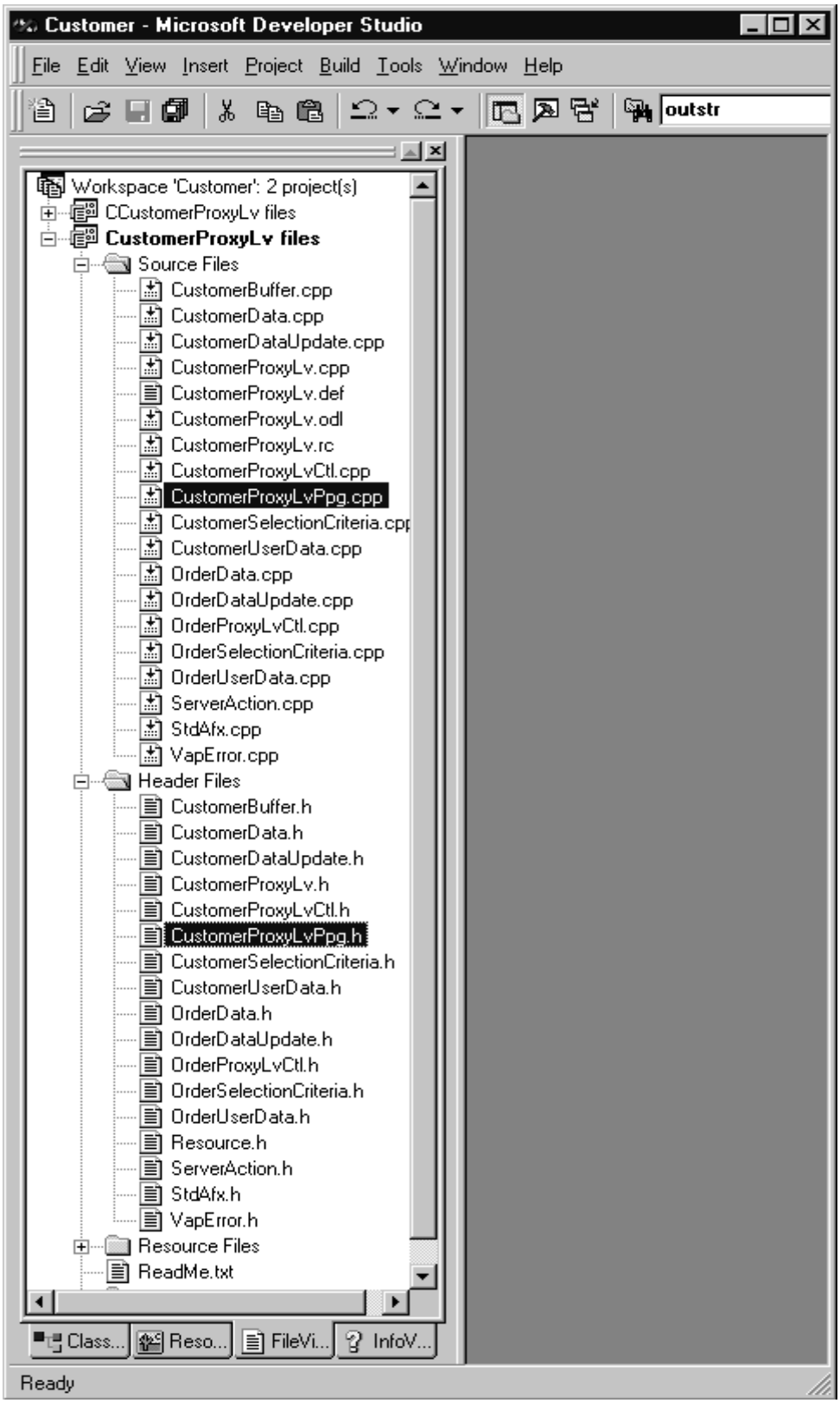
- Cliquez sur le menu **Project** de la barre d'outils puis dans le sous-menu **Add To Project** puis **Files…**
- Sélectionnez tous les fichiers dans le répertoire du projet *Folder2* et cliquez sur **OK**.



Vous devez maintenant effectuer la même opération pour le projet de la Proxy.

- Cliquez sur le menu **Project** de la barre d'outils et sélectionnez le sous-menu **Set Active Project** dans le menu déroulant.
- Sélectionnez le projet de la librairie statique **<nomNoeudRacine>ProxyLv**
- Cliquez sur le menu **Project** de la barre d'outils puis dans le sous-menu **Add To Project** puis **Files…**
- Sélectionnez tous les fichiers dans le répertoire du projet, puis cliquez sur **OK**.

- Supprimez ensuite les deux fichiers dont la terminaison est respectivement **<nomNoeudRacine>Ppg.h** et **<nomNoeudRacine>Ppg.cpp**. du projet.



4.5.6. Paramétrage des chemins d'accès pour la compilation de la Proxy

Vous devez maintenant spécifier les chemins d'accès aux fichiers d'inclusions, librairies, et DLLs.

- Cliquez sur le menu **Tools** de la barre d'outils et choisissez le sous-menu **Options**.
- Dans le notebook, cliquez sur l'onglet **Directories.**
- Dans la liste déroulante **Show directories for**, sélectionnez **Include files**, puis déplacez-vous en bas de la liste **Directories**
- Entrez le chemin complet du répertoire où se trouvent les fichiers à inclure (par défaut à l'installation **d:\vapb\ActiveX\Include**)
- Entrer le chemin du répertoire du projet de la librairie statique : **…<nomWorkspace>\C<nomNoeudRacine>ProxyLv**
- Dans la liste déroulante **Show directories for**, sélectionnez **Library files**, puis déplacez-vous en bas de la liste **Directories.**
- Entrez le chemin complet du répertoire où se trouvent les fichiers de liens (par défaut à l'installation **d:\vapb\ActiveX\Lib**).
- Dans la liste déroulante **Show directories for**, sélectionnez **Executable files**, puis déplacez-vous en bas de la liste **Directories.**
- Entrez le chemin complet du répertoire où se trouvent les DLLs nécessaires (par défaut à l'installation **d:\vapb\ActiveX\Bin**).

4.5.7. Compilation de la librairie statique

Le projet obtenu est maintenant compilable pour obtenir la librairie de sortie nécessaire à la Proxy COM.

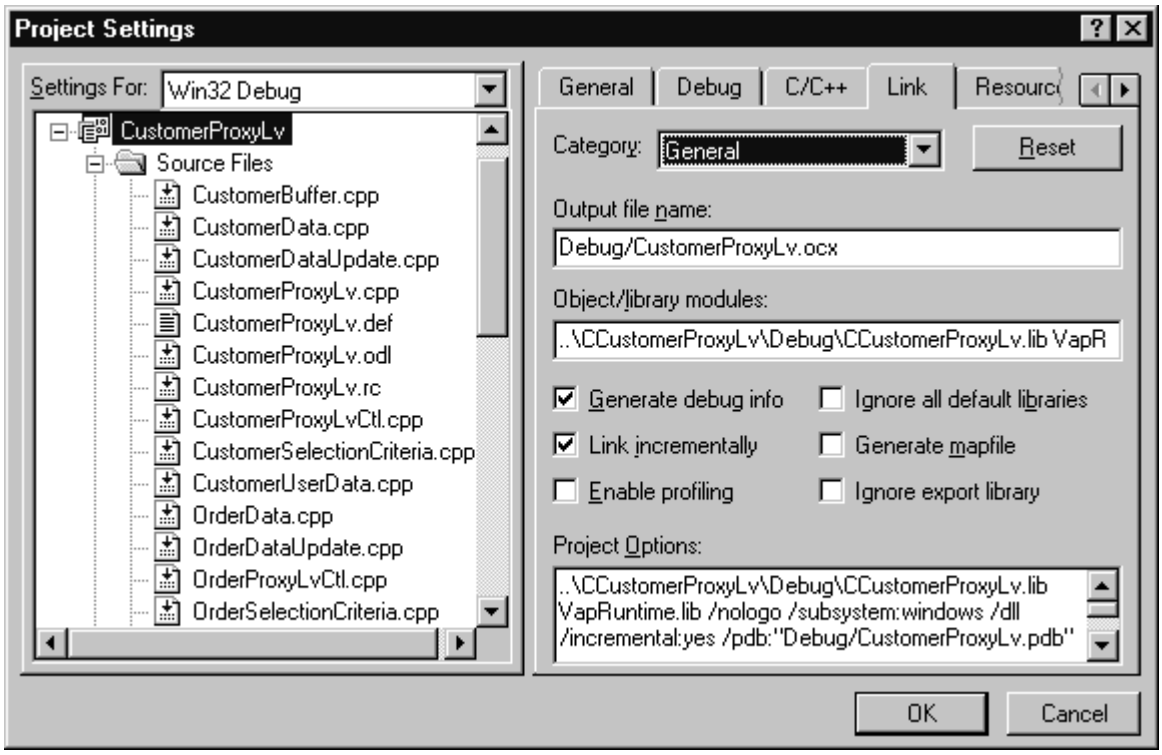
- Cliquez sur le menu **Project** de la barre d'outils et sélectionnez le sous-menu **Set Active Project** dans le menu déroulant.
- Sélectionnez le projet de la proxy **C<nomNoeudRacine>ProxyLv**
- Cliquez sur le menu **Build** de la barre d'outils, puis sélectionnez **Build C<nomNoeudRacine.lib>** dans le menu déroulant, ou utilisez la touche fonction F7. Par défaut la compilation se fait en mode WIN32 Debug et le répertoire de sortie se trouve sous **\<NomWorkspace\C<nomNoeudRacine>ProxyLv\Debug**.

4.5.8. Paramétrage pour l'édition des liens

Vous devez maintenant spécifier les liens entre l'objet et les fichiers **.lib** des librairies utilisées.

- Cliquez sur le menu **Project** de la barre d'outils et choisissez le sous-menu **Set Active Project** dans le menu déroulant.

- Sélectionnez le projet de la Proxy **<nomNoeudRacine>ProxyLv**.
- Cliquez sur le menu **Project** de la barre d'outils et choisissez le sous-menu **Settings** dans le menu déroulant.
- Dans le notebook, cliquez sur l'onglet **Link.**
- Dans la zone **Object/library modules**, saisissez le nom des librairies à linker avec la proxy ainsi que la librairie précédemment obtenue par la compilation du premier projet, soit : **..\C<nomNoeudRacine>ProxyLv\Debug\C<nomNoeudRacine>ProxyLv.lib VapRuntim.lib**
- Validez par **OK**.



4.5.9. Compilation de la Proxy

Cliquez sur le menu **Build** de la barre d'outils et sur **Build <nomNoeudRacine.ocx>** dans le menu déroulant, ou utilisez la touche fonction F7. Le répertoire de sortie se trouve sous **\<NomWorkspace\<nomNoeudRacine>ProxyLv\Debug**.

5. Développement d'un Client VisualAge Java

Après avoir généré les objets Proxy puis les avoir importés dans la station VisualAge, il ne vous reste plus qu'à les intégrer dans l'application graphique.

Après présentation des principes généraux, ce chapitre détaille pas à pas cette étape fondamentale : insertion des objets Proxy avec les liens de programmation impliquant les méthodes, propriétés et événements, gestion des erreurs, gestion de la communication et enfin test de l'application.

Pour faciliter le développement et la maintenance des clients développés dans VisualAge, il est conseillé d'utiliser un projet par application fonctionnelle. Ledit projet doit au moins contenir un ou plusieurs packages de classes graphiques et un package de classes générées.

5.1. Principes généraux

☞ Si votre Proxy ne contient qu'une seule Proxy Elémentaire (nécessairement une Proxy Racine), les propriétés, méthodes et événements associés aux lectures massives ou aux nœuds références ou dépendants ne sont pas disponibles.

5.1.1. Représentation visuelle des objets Proxy dans le Composition Editor

Une fois importée dans la Station VisualAge, la Proxy Vue de Dossier est utilisable par un bean graphique.

Sont présentées ci-dessous les icônes correspondant aux différents objets Proxy, fournies à l'installation.

Icônes	Types de Proxy Elémentaire possibles
	Proxy Racine
	Proxy Dépendante 0,N
	Proxy Dépendante 0,1
	Proxy Dépendante 1,N
	Proxy Dépendante 1,1
	Proxy Référence 0,1
	Proxy Référence 1,1

5.1.2. Utilisation des propriétés

Une propriété correspond à une information gérée par un objet Proxy. Cette information définit une donnée élémentaire, une liste de données élémentaires ou une liste d'instances de données composées. Une propriété peut correspondre à une constante, à un paramètre ou à un résultat de méthode. En fonction du contexte, il est initialisé par l'application graphique ou la Proxy.

On distingue deux types de propriétés :

- celles qui représentent des variables technologiques. Ils permettent d'affiner le comportement des composants proxy dans VisualAge.
- celles qui correspondent aux données de la Vue Logique.

La disponibilité d'une propriété est fonction du type de Proxy. Toutes les propriétés de l'interface publique sont documentées dans le *Manuel de Référence Clients Graphiques : Interface Publique des composants générés*.

5.1.2.1. Contrôles locaux

La Proxy Elémentaire exécute automatiquement les contrôles locaux lors de la création et de la modification d'une instance, via les méthodes **`createInstance`** et **`modifyInstance`**. Chaque Rubrique appartenant à la Vue Logique est contrôlée.

Les contrôles exécutés sont les suivants :

- Contrôles des listes de valeurs définies dans la description des Rubriques.
- Contrôles des intervalles définis dans la description des Rubriques.
- Contrôles de présence obligatoire définis au niveau de l'appel des Rubriques dans une Vue Logique. La présence des Rubriques de type identifiant ou de type foreign key pour une relation référence de cardinalité minimum de 1 est contrôlée automatiquement.

Si les contrôles locaux détectent une erreur, un message d'erreur est positionné via le Gestionnaire d'erreurs.

Ces contrôles peuvent être déclenchés de manière sélective pour chaque noeud de type racine ou dépendant du Dossier concerné. L'attribut permettant d'activer le contrôle des Rubriques sur le serveur est **`serverCheckOption`**.

Que l'on soit en émission ou réception de message, la détection d'une Rubrique vide est automatique.

En revanche, la Proxy n'assure pas les contrôles de numéricité et de date, qui sont pris en charge par les contrôles graphiques.

5.1.2.2. Contrôle de longueur des champs de la propriété detail

Pour chaque champ de la propriété **`detail`**, lors des contrôles effectués pour une création ou une modification locale d'instance, la longueur de la valeur contenue dans la propriété ne doit pas dépasser la longueur maximale de la valeur de cette propriété.

Le contrôle s'effectue systématiquement (sauf si la propriété n'appartient pas au sous-schéma courant) même si la propriété a été définie comme "non à contrôler dans le client".

Une erreur locale est envoyée en cas de longueur excessive.

Il n'y a pas de contrôle de longueur sur les champs inclus dans les buffers utilisateur. Si la longueur est excessive, elle est tronquée à la longueur maximale.

5.1.2.3. Sélection du tri local ou serveur sur une liste d'instances

La propriété **`localSort`** permet de spécifier si la Proxy trie les instances de la propriété **`rows`** à partir du tri défini en local (**`true`**) ou laisse les instances triées dans l'ordre d'envoi du serveur (**`false`**).

Vous pouvez à tout moment modifier le type de tri.

☞ Cette propriété n'est pas effectif dans les services utilisateur.

5.1.2.3.1. Tri local

Le tri local des instances de la propriété **`rows`** correspond au fonctionnement standard de la Proxy si le paramétrage n'a pas été modifié après sa génération. Dans ce contexte, deux types de tri sont possibles :

- Si aucun critère de tri n'est défini en local (voir paragraphe **5.1.2.4**), la Proxy trie implicitement les instances dans l'ordre croissant des identifiants définis sur la Vue Logique.
- Si un critère de tri est défini en local, la Proxy trie les instances dans l'ordre défini par ce dernier.

Dans tous les cas, la création locale d'une instance insère celle-ci en fonction du critère de tri courant appliqué dans la propriété **`rows`**.

Le changement dynamique ou la suppression du critère de tri local induit immédiatement un tri sur les instances contenues dans la propriété **`rows`**.

5.1.2.3.2. Tri serveur

Le tri serveur des instances de la propriété **`rows`** est déclenché si la propriété **`localSort`** est à **`false`**. Dans ce contexte, les instances contenues dans la propriété **`rows`** sont présentées dans l'ordre dans lequel elles ont été reçues du serveur, quel que soit le contenu du critère de tri défini en local.

Dans le contexte de gestion manuelle des collections ou de pagination en mode extend, les instances reçues sont ajoutées en fin de la collection existante dans la propriété **`rows`**.

Toute instance créée localement est systématiquement ajoutée à la fin de la collection existante dans la propriété **`rows`**. Dans ce contexte, une instance qui n'est pas positionnée en fin d'une collection qui est supprimée et recréée localement est transférée à la fin de la collection contenue dans la propriété **`rows`**.

5.1.2.4. Spécification du critère de tri local

Il est possible de modifier dynamiquement le critère de tri utilisé pour présenter les instances dans la propriété **`rows`**, en utilisant la propriété **`dataComparator`** de chaque Proxy : **`void setDataComparator(Comparator c)`**.

Le paramètre requis est une instance d'une classe implémentant l'interface **`com.ibm.vap.generic.Comparator`**.

Cette interface se compose d'une méthode représentant la relation d'ordre suivante : **`int compare(Object a, Object b)`**.

Cette méthode doit renvoyer :

- un nombre négatif si a < b
- 0 si a = b

- et un nombre positif si a > b.

Par exemple :

```
import com.ibm.vap.generic.Comparator ;
public final class CustomerComparator implements Comparator {
public static final int NAME = 0 ;
public static final int COMPANY = 1 ;
public int criteria ;
public int compare(Object a, Object b) {
try {
switch(criteria) {
case NAME:
return ((CustomerData)a).getName().compareTo(
(CustomerData)a).getName());
break;
case COMPANY:
return ((CustomerData)a).getComp().compareTo(
(CustomerData)a).getComp());
break;
                }
} catch (IllegalCastException ice) {
return 0;
         }
  }
}
```

5.1.2.5. TableModel

Cette propriété est disponible en lecture/écriture sur tous les types de nœud si vous avez choisi l'option de génération **`Utiliser Swing`** (disponible à partir de la version 2 de VisualAge Java).

Cette propriété permet d'intégrer dans une application une JTable, composant swing constitué de plusieurs lignes et de plusieurs colonnes.

Par défaut, cette propriété est initialisée avec une nouvelle instance de TableModel générée.

5.1.2.6. Gestion des sous-schémas

Les sous-schémas spécifiés dans la description de la Vue Logique peuvent être pris en compte par les méthodes de sélection/lecture si les Composants Applicatifs gèrent la présence des Rubriques (options **`VECTPRES=YES`** ou **`CHECKSER=YES`**).

Chaque nœud possède deux propriétés :

- **`subSchema`**, qui permet d'attribuer le sous-schéma désiré lors d'une sélection, lecture ou mise à jour du Composant Applicatif du nœud. Cette propriété peut être alimentée à partir de la propriété **`subSchemaList`**.
- **`subSchemaList`**, qui permet de lister tous les sous-schémas disponibles pour le nœud. Comme il n'est pas possible dans le Référentiel VisualAge Pacbase d'affecter un nom au sous-schéma, chaque sous-schéma est désigné par **`SubSchema<n>`** (avec **`n`** de 01 à 10).

5.1.3. Utilisation des méthodes

5.1.3.1. Mise en œuvre

Une méthode correspond à un traitement qu'un objet Proxy peut exécuter. Elle est déclenchée à l'aide d'une connexion entre un événement de l'application graphique et le code d'une méthode d'une Proxy.

La disponibilité d'une méthode est fonction du type de Proxy. Toutes les méthodes de l'interface publique sont documentées dans le manuel *Manuel de Référence Clients Graphiques : Interface Publique des composants générés.*

5.1.3.2. Les différents types de méthodes serveur

Les méthodes serveur exécutent des traitements implémentés dans un ou plusieurs Composants Applicatifs associés au Dossier. Ces méthodes émettent une requête vers les Composants Applicatifs qui renvoient un résultat sur la station de travail. Les requêtes et les réponses contiennent généralement des paramètres techniques, des instances de Vue Logique associées à un ou plusieurs noeuds et des informations contextuelles définies dans un buffer utilisateur.

Il faut distinguer deux types de méthodes serveur.

- celles qui accèdent systématiquement au serveur :
  - **`selectInstances`**,
  - **`readInstance`**,
  - **`readInstanceAndLock`**,
  - **`readInstanceWithFirstChildren`**,
  - **`readInstanceWithAllChildren`**,
  - **`readInstanceWithFirstChildrenAndLock`**,
  - **`readInstanceWithAllChildrenAndLock`**,
  - **`readAllChildrenFromDetail.`**
  - **`readAllChildrenFrom`**
- celles qui ne font pas un accès systématique au serveur :
  - **`readNextPage`** : il y a accès au serveur sauf si lors de la précédente sélection, l'événement **`noPageAfter`** a été renvoyé.
  - **`readPreviousPage`** : il y a accès au serveur sauf si lors de la précédente sélection, l'événement **`noPageBefore`** a été renvoyé.
  - **`readFirstChildrenFromDetail`** : il y a accès au serveur, sauf si la propriété **`maximumNumberOfRequestedInstances`** des Proxy Dépendantes est positionnée à 0 et si la propriété **`globalSelection`** est positionnée à false.
  - **`readFirstChildrenFrom`** : il y a accès au serveur, sauf si la propriété **`maximumNumberOfRequestedInstances`** des Proxy Dépendantes est positionnée à 0 et si la propriété **`globalSelection`** est positionnée à false.
  - **`checkExistenceOfDependentInstances`** : il y a accès au serveur sauf s'il existe en local la possibilité de vérifier l'existence d'instances dépendantes.
  - **`updateFolder`** : il n'y a d'accès au serveur que s'il existe au moins une instance du noeud concerné modifiée dans sa propriété **`updatedFolders.`**

5.1.3.3. Gestion des lectures d'un Dossier

La lecture massive de la racine d'un Dossier permet de lire sur un composant client toutes les occurrences du noeud racine du Dossier présentes dans la base de données. Les méthodes concernées sont **`selectInstances`** et **`readNextPage`**.

5.1.3.3.1. Lecture massive par anticipation des noeuds dépendants

Dans la cinématique des architectures client/serveur, une application graphique manipulant un Dossier cherche à acquérir des informations par anticipation pour minimiser les échanges avec les serveurs.

Dans un réseau hiérarchique, plusieurs méthodes d'anticipation des lectures peuvent être envisagées :

- La première méthode de type '**`allChildren`**' lit *toutes* les instances dépendantes de l'instance sélectionnée dans la propriété **`detail`** de la Proxy Elémentaire parente.
- La deuxième méthode de type '**`firstChildren`**' ne lit que les instances *immédiatement* dépendantes de l'instance sélectionnée dans la propriété **`detail`** de la Proxy Elémentaire parente.

La première méthode de lecture massive par anticipation n'est disponible que sur la Proxy Racine.

La deuxième méthode de lecture massive par anticipation est disponible sur la Proxy Racine ou sur les Proxy Dépendantes qui possèdent également des Proxy Dépendantes.

5.1.3.3.2. Transfert d'instance entre les propriétés rows et detail

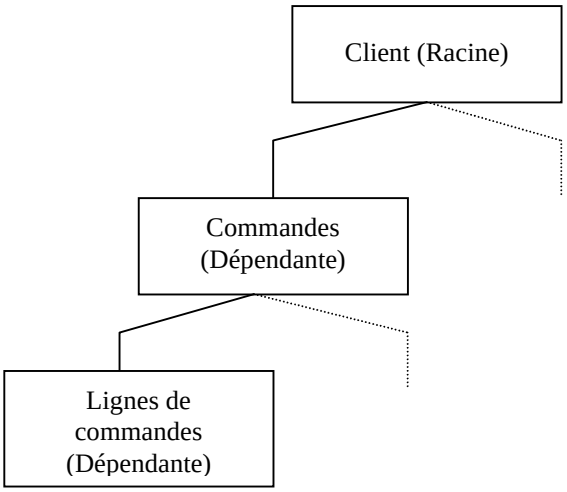
Le transfert d'instance entre la propriété **`rows`** et **`detail`** permet d'alimenter la propriété **`detail`** avec une instance initialement lue par une méthode de lecture massive.

Ce transfert n'est disponible que sur les Proxy Racines et Dépendantes. Il correspond à une méthode de lecture locale qui alimente également toutes les instances locales des Proxy Dépendantes connues par la Proxy Vue de Dossier. Le transfert est réalisé par l'intermédiaire de la méthode **`getDetailFromDataDescription.`**

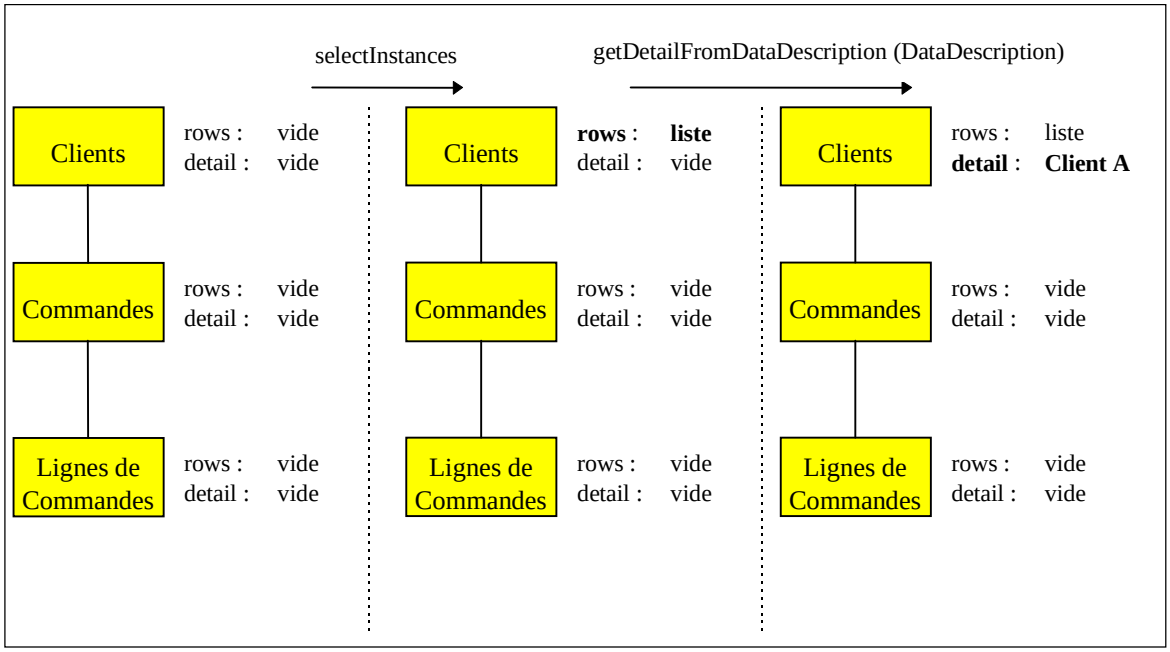
5.1.3.3.3. Lecture massive et transfert d'instance entre les propriétés rows et detail : cinématique de fonctionnement

Cet exemple illustre l'alimentation de **`detail`** avec une instance préalablement transférée dans **`rows`** par une méthode de lecture massive d'un noeud racine ou dépendant et le principe de la lecture massive par anticipation des noeuds dépendants.

Il est basé sur une Proxy Vue de Dossier composée de trois Proxy Elémentaires :

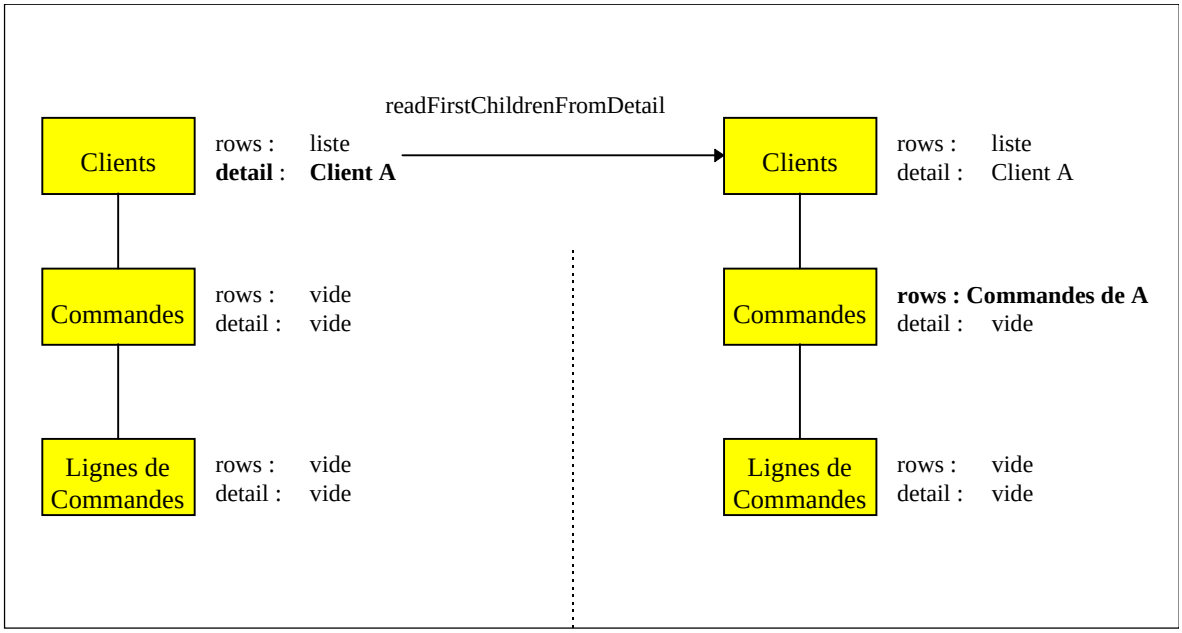


Les schémas ci-dessous présentent la cinématique de fonctionnement basé sur ce principe.



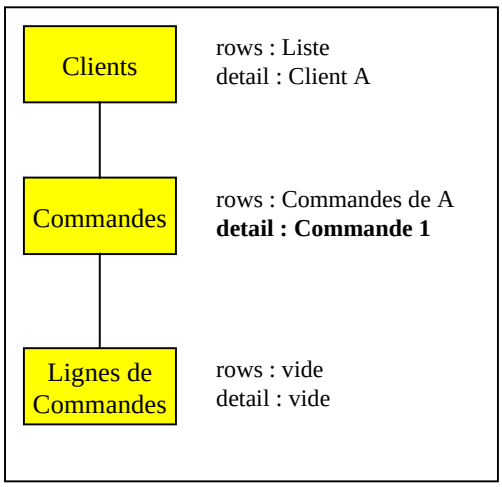
La propriété **`detail`** du noeud Clients contient à présent le Client A. Ensuite, pour lire les instances dépendantes du Client A, c'est-à-dire ses commandes, vous disposez de trois solutions.

SOLUTION 1



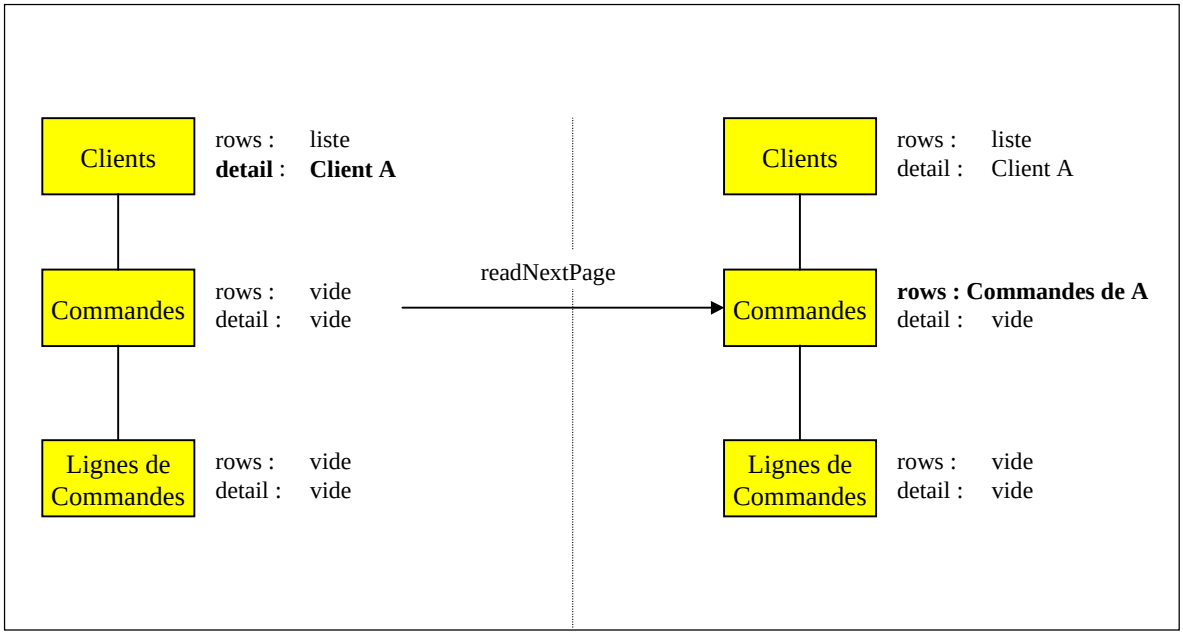
La méthode **`readFirstChildrenFromDetail`** sur la Proxy Racine Clients alimente non seulement la propriété **`rows`** de la Proxy Dépendante Commandes pour le Client A mais aussi la propriété **`rows`** des éventuelles autres Proxy Elémentaires directement dépendantes de la Proxy Racine.

Dans ce contexte, la méthode **`getDetailFromDataDescription (DataDescription)`** sur la Proxy Dépendante Commandes provoque :



Pour lire ensuite les lignes de commandes de la Commande 1 du Client A, utilisez la méthode **`readFirstChildrenFromDetail`** sur Commandes ou la méthode **`readNextPage`** sur Lignes de commandes.

SOLUTION 2

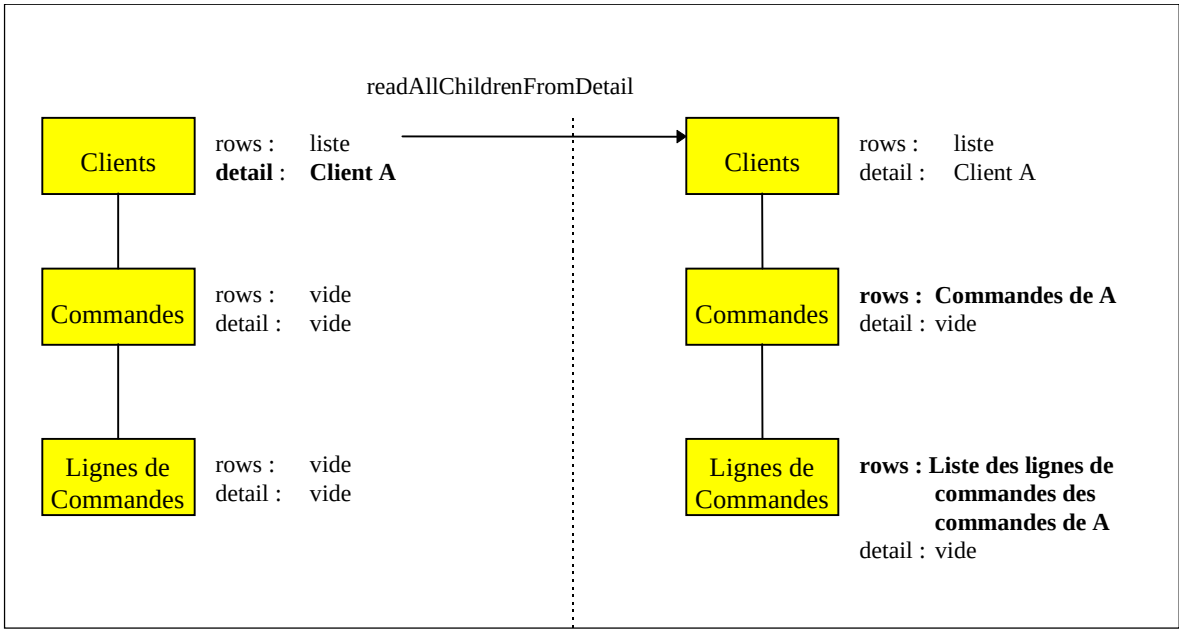


La méthode **`readNextPage`** sur la Proxy Dépendante Commandes alimente sa propriété **`rows`**. Les instances des éventuelles autres Proxy Elémentaires directement dépendantes de la Proxy Racine ne sont pas lues.

Dans ce contexte, la méthode **`getDetailFromDataDescription (DataDescription)`** sur la Proxy Dépendante Commandes provoque le même résultat que dans la solution 1.

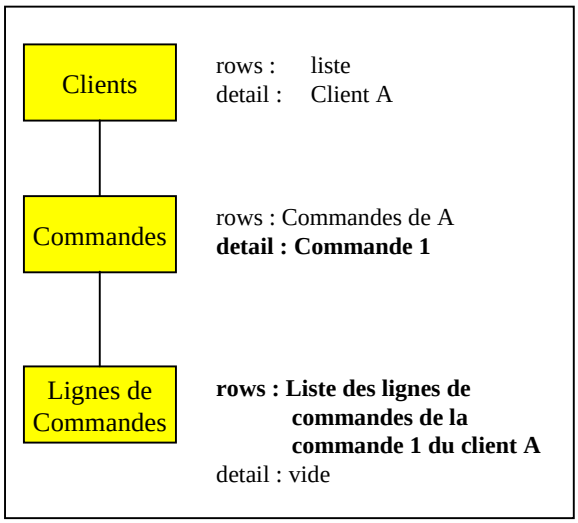
Pour lire ensuite les lignes de commandes de la Commande 1 du Client A, il faut procéder de la même manière que dans la solution précédente.

SOLUTION 3



La méthode **`readAllChildrenFromDetail`** sur la Proxy Racine Clients lit non seulement toutes les commandes de A mais aussi toutes les éventuelles autres instances dépendantes de A quelque soit leur niveau hiérarchique : ainsi dans notre exemple, toutes les lignes de commandes de toutes les commandes de A sont lues.

Dans ce contexte, la méthode **`getDetailFromDataDescription (DataDescription)`** sur la Proxy Dépendante Commandes provoque :



La méthode **`getDetailFromDataDescription (DataDescription)`** alimente automatiquement la propriété **`rows`** de la Proxy Dépendante Lignes de commandes avec les lignes de commandes de la Commande 1 du Client A, ces lignes ayant été transférées en local au préalable par la méthode **`readAllChildrenFromDetail`**.

5.1.3.3.4. Lecture massive des noeuds références

La lecture d'un noeud référence est considérée comme une aide sur critères. Elle permet de montrer à l'utilisateur final une liste d'informations potentiellement référençables sur un noeud dépendant.

Les informations présentées à l'utilisateur correspondent aux informations nécessaires et suffisantes pour assister l'utilisateur final dans son choix.

Pour optimiser le volume des caractères transmis pour ce type de service, les Vues Logiques disposent d'un sous-schéma de type « aide sur critères » qui permet de sélectionner les Rubriques concernées au niveau du serveur.

Les lectures massives des noeuds références sont exécutées sur demande et ne peuvent pas être intégrées dans le traitement des lectures par anticipation. Les méthodes concernées sont `selectInstances` et `readNextPage`.

5.1.3.3.5. Principe de pagination dans les noeuds d'un Dossier

Deux types de pagination sont proposés sur les noeuds d'un Dossier :

- Le premier type de pagination, appelé pagination en mode *non-extend*, permet de paginer en avant et en arrière sur une collection prédéfinie, par l'intermédiaire de méthodes spécifiques. Chaque méthode exécute une demande de lecture au serveur et son résultat remplace le résultat de la lecture précédente. Ce type de pagination n'est applicable que sur les noeuds racines ou références.
- Le deuxième type de pagination, appelé pagination en mode *extend*, permet de lire graduellement les instances d'une collection définie, au fur et à mesure des demandes de lectures des pages suivantes. Dans ce contexte, la lecture des pages précédentes disparaît et cette fonction est reprise en local par les ascenseurs du contrôle graphique qui présente la liste des instances. Ce type de pagination est applicable à tous les noeuds d'un Dossier.

5.1.3.3.6. Critères de sélection associés aux lectures massives

Les critères de sélection associés aux lectures massives sont des propriétés élémentaires ou composées associées à chaque noeud d'un Dossier. Ils se répartissent en deux types :

- Les critères de sélection fonctionnels correspondent à l'identifiant et aux éléments nécessaires à la définition des méthodes d'extraction de la Vue Logique associée au noeud. Ces critères correspondent aux propriétés suivantes :
  - `selectionCriteria` définit les Rubriques de type identifiant et paramètres par valeur.
  - `extractMethodCode` définit le code de la méthode d'extraction souhaitée.
  - `extractMethodCodes` contient la liste des méthodes d'extraction disponibles.

- Les critères de sélection organiques correspondent aux informations permettant de maîtriser le volume des instances sélectionné sur chaque nœud.

- **globalSelection** est une propriété booléenne qui, lorsqu'elle est positionnée à true, permet de lire la totalité des instances du noeud sur une requête de sélection.
- **maximumNumberOfRequestedInstances** est une propriété numérique qui indique, lorsque la propriété **globalSelection** est à false, le nombre d'instances d'un noeud à lire sur une requête de sélection. Cette propriété peut contenir la valeur 0. Dans ce cas, la branche n'est pas lue lors d'une lecture massive par anticipation.

5.1.3.3.7. Limitation du champ des lectures massives

Lors d'une lecture massive avec utilisation de critères de sélection, seules les instances qui correspondent à ces critères sont lues.

Cependant, dans le cas d'une lecture massive commandée par une lecture par anticipation de type **allChildren**, on considère que la propriété **globalSelection** est positionnée à true sur chaque noeud.

5.1.3.3.8. Lecture d'une instance d'un noeud racine ou dépendant

La lecture d'une instance d'un noeud racine ou dépendant permet d'alimenter directement la propriété **detail** d'un noeud sans passer au préalable par une sélection massive d'une collection des instances de ce noeud.

Ce type de lecture est considéré comme une sélection de collection et annule donc la sélection précédente même si celle-ci correspondait à une lecture massive. L'alimentation de la propriété **detail** entraîne donc l'initialisation de la propriété **rows**.

Les méthodes qui implémentent cette fonction de sélection permettent :

- Une lecture de l'instance sans ses dépendances (**readInstance**).
- Une lecture de l'instance avec ses dépendances de premier niveau (**readInstanceWithFirstChildren**).
- Une lecture de l'instance avec toutes ses dépendances (**readInstanceWithAllChildren**).

5.1.3.3.9. Lecture d'une instance d'un noeud référence

La lecture d'une instance d'un noeud référence ne peut pas activer de processus de lecture massive par anticipation. Elle permet de lire la totalité de la description de l'instance du noeud appelé dans sa propriété **detail**.

Seule la méthode **readInstance** est donc disponible sur un noeud référence.

Les noeuds références sont dépourvus de méthode de mise à jour. Leurs propriétés **rows** et **detail** sont indépendantes. Le résultat de la méthode **readInstance** n'initialise donc pas la propriété **rows**.

La propriété **rows** permet de visualiser les informations suffisantes pour permettre d'affecter une des instances du noeud référence à l'instance du noeud référençant.

La propriété **detail** permet de consulter la totalité de la description d'une instance référencée.

La structure de ces deux propriétés peut donc être différente.

5.1.3.3.10. Critères de sélection associés aux lectures d'une instance

L'identifiant de l'instance du noeud à lire est spécifié dans les critères de sélection fonctionnels associés au noeud.

Pour les noeuds dépendants, l'identifiant du noeud correspond à l'identifiant de la Vue Logique associée au noeud débarrassé des Rubriques définissant les identifiants des Vues Logiques hiérarchiquement supérieures. Ces identifiants hiérarchiques sont initialisés automatiquement par la Proxy Vue de Dossier en fonction de la navigation dans le Dossier.

5.1.3.4. Gestion des mises à jour d'un Dossier

5.1.3.4.1. Mises à jour locales

Les services de mise à jour locales sont disponibles sur chaque noeud de type racine ou dépendant du Dossier.

- Création d'une instance de noeud,
- Modification d'une instance de noeud,
- Annulation d'une instance de noeud.

Cependant, il existe d'autres règles inhérentes à la gestion des Dossiers :

- La création d'une instance d'un noeud dépendant n'est autorisée que si la hiérarchie des instances contenues dans les propriétés `detail` des noeuds supérieurs existe.
- L'annulation d'une instance d'un noeud entraîne l'annulation récursive des instances locales des noeuds dépendants.

Pour permettre au développeur de gérer des messages permettant d'avertir l'utilisateur de l'impact d'une annulation en cascade, une méthode de vérification d'existence d'instances dépendantes est disponible sur les noeuds de type racine ou dépendant.

Cette méthode (`checkExistenceOfDependentInstances`) émet un résultat booléen true ou false. Si aucune dépendance n'est trouvée dans le cache local du Dossier et que l'instance concernée n'a pas été créée localement, elle émet une requête de vérification sur le serveur.

Pour qu'un utilisateur puisse revenir en arrière sur les manipulations locales d'une instance de Dossier, une méthode `undoAllLocalFolderUpdates` permet d'éliminer toutes les mises à jour locales sur tous les noeuds du Dossier appliquées depuis la dernière exécution d'une méthode de mise à jour serveur. Cette méthode n'est disponible que sur le noeud racine du Dossier. Une autre méthode `undoLocalFolderUpdates` permet d'éliminer toutes les mises à jour associées à l'instance du nœud de Racine que vous aurez paramétré.

5.1.3.4.2. Mises à jour serveur

La Proxy Racine est seule à disposer des méthodes de mise à jour serveur.

Les mises à jour serveur correspondent aux méthodes permettant à un composant client d'envoyer au serveur l'ensemble des mises à jour locales effectuées depuis la dernière mise à jour de ce type.

Ces mises à jour concernent toutes les instances dépendantes modifiées. Elles peuvent concerner plusieurs instances de Dossier.

Lorsqu'une méthode de mise à jour serveur renvoie des erreurs, elle laisse le Dossier en statut « modifié localement » et seule la correction des erreurs et le renvoi des mises à jour, ou de la méthode **`undoAllLocalFolderUpdates`** ou **`undoLocalFolderUpdates`** permettent d'effectuer de nouvelles sélections de collection.

Les méthodes de mise à jour serveur exécutent, avant l'émission de la requête sur le serveur, un contrôle de l'intégrité du Dossier pour les instances créées localement. Cette méthode contrôle, pour chaque instance de noeud créé en local, les cardinalités minimum de chaque lien et émet une erreur si le nombre des instances dépendantes ne respecte pas les propriétés des liens associés.

Une méthode de mise à jour serveur peut être accompagnée d'une demande de rafraîchissement des instances mises à jour dans le cas où certaines Rubriques de ces instances, comme les identifiants, sont calculées par le serveur. Cette demande de rafraîchissement passe par l'utilisation de la propriété **`refreshOption`**.

5.1.3.4.3. Gestion des mouvements utiles

La gestion des mouvements utiles, assurée par le cache local, est entièrement automatique et transparente pour le développeur.

Elle 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. Si plusieurs mises à jour locales ont été effectuées sur une même instance de noeud, seule la dernière sera envoyée au serveur.

5.1.3.4.4. Changement de sélection d'une collection

L'événement **`aboutToChangeSelection`** est envoyé au Client par la Proxy Racine lorsqu'une méthode de sélection serveur sur un noeud du Dossier remplace sa collection locale courante et que cette collection ou les collections dépendantes contiennent des modifications locales candidates à une mise à jour serveur. Si l'événement n'est pas intercepté lors de la programmation, les mises à jour sont perdues.

5.1.3.4.5. Réinitialisation des instances du cache local

La méthode **`resetCollection`** permet d'éliminer explicitement toutes les instances contenues dans le cache d'une Proxy Vue de Dossier avant de reconstituer une nouvelle collection. Cette méthode est disponible sur tous les types de nœuds d'une Proxy Vue de Dossier disposant de la propriété **`Rows`**.

5.1.3.4.6. Gestion des collections d'instances

La gestion des collections d'instances peut se faire soit automatiquement, soit manuellement par le positionnement de la propriété booléenne **`manualCollectionReset`** disponible sur tous les types de nœuds d'une Proxy Vue de Dossier. Le mode de gestion manuel permet de constituer des collections hétérogènes par une succession de méthodes de sélection de collections et de pagination.

5.1.3.4.7. Peuplement du cache local sans accès serveur

La méthode **`initializeInstance`** permet de stocker dans le cache local une instance de Vue Logique non lue sur le serveur et qui n'a pas été créée localement. Elle permet donc d'envoyer la mise à jour de l'instance de Vue Logique sans qu'elle ait été lue prélablement sur le serveur. Cette méthode est disponible sur tous les types de nœuds et est valide dès lors que l'instance de Vue Logique n'existe pas en local.

5.1.3.5. Méthodes asynchrones

5.1.3.5.1. Principes

La programmation asynchrone permet de dissocier la méthode d'émission de la requête de celle qui permet de récupérer sa réponse. Vous pouvez utiliser ce type de programmation que vous utilisiez un protocole de communication asynchrone ou non.

Il est possible d'utiliser les composants Proxy en mode asynchrone, et ceci de façon totalement indépendante du middleware utilisé. Ainsi, on peut travailler en mode asynchrone au niveau Proxy, en utilisant une *location* dont le middleware est synchrone, et réciproquement.

Dans ce contexte, l'utilisateur final améliore sa productivité en soumettant à l'avance des requêtes dont il récupèrera la réponse au moment où il en aura besoin. Cette technique de travail par anticipation supprime la perception négative de l'inactivité associée aux temps de réponse.

De plus, les protocoles de communication permettent de sécuriser les requêtes ou les réponses dans des files de messages locales en garantissant l'acheminement du message quel que soit l'état du réseau.

Le mode de conversation est défini par la propriété booléenne **`Asynchronous`** au niveau Dossier. Ainsi :

```
myFolder.setAsynchronous(true);
```

provoque le passage en mode asynchrone.

Par la suite, toute méthode faisant accès au serveur provoque une exception **`com.ibm.vap.generic.AsynchrounousRequestException`**.

Cette exception ne possède qu'une méthode publique :

```
com.ibm.vap.generic.ServerActionContext getContext();
```

L'instance de **`ServerActionContext`** renvoyée doit être conservée. Elle constitue la clé permettant de traiter ultérieurement la réponse du serveur, par la méthode **`getReply()`** :

```
boolean b = myFolder.getReply(context);
```

Il n'est pas toujours possible de traiter la réponse associée à un contexte de méthode asynchrone, et ceci pour deux raisons potentielles :

- La réponse peut ne pas être encore parvenue (la méthode **`getReply()`** renvoie false)
- Le contexte en question n'est plus valide, car il était associé à une instance qui a disparu du cache local (une exception locale est levée).

Dans le cas où aucune erreur n'est survenue et que la réponse a pu être traitée, **`getReply(context)`** renvoie true.

5.1.3.5.2. Méthodes globales ou associées à une instance

Certaines méthodes serveur, désignées comme méthodes globales, sont indépendantes de toute sélection, alors que d'autres dépendent d'une instance donnée contenue dans le cache local. En mode asynchrone, les méthodes globales stockent les identifiants de réponse dans une collection et chaque requête exécutée ajoute son identifiant dans cette collection ; les méthodes associées à une instance de Vue Logique stockent les identifiants de réponse dans une collection associée à l'instance concernée. Les collections d'identifiants de réponse associées aux méthodes globales sont perdues lorsque l'application qui utilise la Proxy est fermée. Une collection d'identifiants de réponse associée à une instance contenue dans le cache local est perdue lorsque cette instance est supprimée localement ou suite à un changement de collection.

- **Méthodes globales**

Les réponses associées aux méthodes suivantes peuvent toujours être traitées, indépendamment de la collection courante :

- `executeUserService ()`
- `readInstance ()` (RAC)
- `readInstanceWithFirstChildren ()` (RAC)
- `readInstanceWithAllChildren ()` (RAC)
- `readInstanceAndLock ()` (RAC)
- `readInstanceWithFirstChildrenAndLock ()` (RAC)
- `readInstanceWithAllChildrenAndLock ()` (RAC)
- `readNextPage ()` (RAC)
- `selectInstances ()`
- `lock ()`
- `unlock ()`
- `readPreviousPage ()`

- **Méthodes associées à une instance**

Les méthodes suivantes dépendent soit de l'instance `detail`, soit de l'instance passée en paramètre, soit de l'instance parente `detail` pour les nœuds dépendants :

- `checkExistenceOfDependentInstances ()`
- `readAllChildren(data)`
- `readFirstChildren(data)`
- `readAllChildrenFromCurrentInstance ()`
- `readAllChildrenFrom{}Data` (DEP)
- `readFirstChildrenFromCurrentInstance ()`
- `readFirstChildrenFrom{}Data` (DEP)
- `readInstance ()` (DEP)
- `readInstanceWithFirstChildren ()` (DEP)
- `readInstanceWithAllChildren ()` (DEP)
- `readInstanceAndLock ()` (DEP)
- `readInstanceWithFirstChildrenAndLock ()` (DEP)
- `readInstanceWithAllChildrenAndLock ()` (DEP)

5.1.3.5.3. Exemples d'utilisation

Vous pouvez utiliser les méthodes asynchrones de deux manières, illustrées ci-après.

- **Polling**

Ce système consiste à « guetter » l'arrivée d'une réponse dans un *thread* pour ne pas bloquer l'application en attendant l'affichage des informations. Le code de la réponse se trouve dans un *thread* distinct de celui de l'application principale, mais possédant un pointeur sur la Proxy Racine. On suppose également qu'il connaît un contexte associé à une méthode asynchrone.

```
while (!myFolder().getReply(context)) {
wait(1000);
}
```

- **Accès en tâche de fond**

L'exemple suivant montre comment procéder pour acquérir des informations sur les instances dépendantes avant que l'utilisateur final ne les demande explicitement, et cela sans que l'application soit bloquée.

- Lorsque l'utilisateur sélectionne une collection d'instances radicales :

```
myFolder.setAsynchronous(false);
myFolder.selectInstances();
```

  ☞ Son résultat étant nécessaire pour la suite, la méthode **`selectInstances`** est utilisée en mode synchrone.

- Ensuite, on va parcourir **`rows`** afin de lire toutes les instances dépendantes de chaque instance lue par la méthode **`selectInstances`**.

  Au préalable, il est nécesaire de passer en mode asynchrone :

```
myFolder.setAsynchronous(true);
Enumeration rows = myFolder.rows.elements();
Vector contexts = new Vector();
while (rows.hasMoreElements()) {
Data currentData = (Data)rows.nextElement();
try {
myFolder.readAllChildren(currentData);
} catch (VapException ve) {
} catch (AsynchronousRequestException are) {
contexts.addElement(are.getContext();
    }
}
myFolder.setAsynchronous(false);
```

- Ensuite, lorsque l'utilisateur décide de travailler sur une donnée data :

```
try {
int index = myFolder.rows().indexOf(data);
    myFolder.getReply(contexts.elementAt(index));
} catch (VapException) {} //Tout s'est bien passé //
myFolder.getDetailFromDataDescription(data);
```

Ainsi, les instances dépendantes apparaissent dans l'arbre de sélection « spontanément ». Il est à noter que les réponses des méthodes ne sont traitées qu'au dernier moment, ce qui n'est pas une obligation.

5.1.3.6. Services utilisateur

Chaque noeud de type racine ou dépendant dispose pour mettre en oeuvre un service utilisateur des éléments suivants :

- Une propriété permettant d'avoir la liste des services utilisateurs disponibles sur ce noeud plus `nil`. Cette propriété est `userServiceCodes`.
- Une propriété permettant d'initialiser le code du service utilisateur à exécuter. Cette propriété est `userServiceCode`.
- Une propriété permettant de stocker localement des instances de Vue Logique à traiter pour le prochain service utilisateur. Cette propriété est `userServiceInputRows`.
- Une propriété permettant de présenter plusieurs instances de Vue Logique renvoyées par un service utilisateur. Cette propriété est `userServiceOutputRows`.
- Une propriété présentant les instances de Vue Logique candidates à l'exécution du prochain service utilisateur. Cette propriété est `userDetail`.
- Des méthodes locales permettant de mémoriser chaque instance de Vue Logique à envoyer au serveur pour l'exécution d'un service utilisateur. Ces méthodes sont `createUserInstance`, `modifyUserInstance` et `deleteUserInstance`.

Le noeud racine du Dossier dispose en plus des éléments suivants :

- Une méthode permettant d'exécuter l'ensemble des services utilisateur paramétrés sur chaque noeud du Dossier. Cette méthode est `executeUserServices`.
- Une méthode permettant d'effacer toutes les instances locales mémorisées pour tous les noeuds du Dossier. Cette méthode est `resetUserServiceInputInstances`.
- Une méthode permettant d'effacer l'instance courante mémorisée en local. Cette méthode est `resetUserServiceCodes`.

Ce principe permet d'exécuter dans la même requête un ensemble de 1 à n services utilisateur répartis sur des noeuds différents. L'ordre d'exécution de ces services correspond à l'ordre hiérarchique des noeuds, en parcourant l'arbre de haut en bas et de gauche à droite.

5.1.3.7. Verrouillage logique de la base

Les mécanismes d'upload-download associés à un Dossier augmentent le temps écoulé entre la lecture d'un Dossier et l'affichage de la mise à jour.

Dans ce contexte, sans mécanisme de verrouillage, deux utilisateurs peuvent modifier la même instance de Dossier. Les mises à jour cumulées deviennent difficiles à gérer.

Pour permettre à l'utilisateur d'utiliser un Dossier dans un mode d'appropriation exclusif, deux types de verrouillage d'une instance de noeud sont à sa disposition :

- Le verrouillage optimiste qui fonctionne sur le principe de la vérification de l'évolution d'un **`TimeStamp`** avant d'exécuter le traitement de mise à jour.
- Le verrouillage pessimiste qui s'approprie en mise à jour exclusive une entité au moyen de l'enregistrement d'une ressource spécifique. Dans ce cas, le traitement de mise à jour du Dossier est effectué avant de libérer la ressource exclusive.

Le traitement de verrouillage serveur est déclenché en fonction de l'exécution explicite d'une méthode spécifique disponible sur la Proxy Racine. Cette méthode est **`lock`**.

Le traitement de déverrouillage serveur est déclenché automatiquement avec l'exécution d'une méthode de mise à jour serveur ou explicitement en fonction de l'exécution d'une méthode spécifique disponible sur la Proxy Racine. Cette méthode spécifique est **`unLock`**.

L'écriture du traitement de verrouillage dans le Composant Applicatif racine est à la charge du développeur.

Ce traitement reçoit l'identifiant de l'instance de Vue Logique à verrouiller ainsi que le type de demande (verrouillage ou déverrouillage) à exécuter.

Il doit positionner en retour un statut permettant d'accorder ou de refuser le verrouillage ainsi que le **`TimeStamp`** ou le nom de la ressource utilisée.

En cas de refus de verrouillage, la Proxy Racine émet un événement **`lockFailed`** et toutes les méthodes de mise à jour locales ou serveur exécutées ultérieurement sont invalidées pour l'instance de Dossier spécifiée. Dans ce cas, le Dossier passe en statut « lecture uniquement ».

Le verrouillage logique est défini dans l'entité Dossier ou dans le Composant Applicatif en cas de développement mono-vue.

Lorsque l'option de verrouillage logique est active sur un Dossier, toute demande de lecture de la propriété **`detail`** de la Proxy Racine peut être accompagnée d'une demande de blocage logique au niveau du serveur.

5.1.3.8. Personnalisation des colonnes d'une JTable

Une JTable est un composant swing disponible à partir de la version 2 de VisualAge Java.

Si vous intégrez ce composant tel quel, il affichera, à l'exécution, toutes les colonnes correspondant à toutes les Rubriques de la Vue Logique, avec les noms en clair définis dans la Vue Logique.

Pour sélectionner les colonnes à afficher, modifier leur en-tête ou encore créer une nouvelle colonne affichant des informations calculées en local, vous devez personnaliser la JTable.

Pour cela, vous devez d'abord créer une nouvelle classe publique héritant soit du TableModel généré (utile pour récupérer une partie de son implémentation) soit directement de **`PacbaseTableModel`** (package **`com.imb.vap.beans.swing`**).

Il suffit ensuite de personnaliser les méthodes suivantes :

- **`public Int getColumnCount()`** : retourne le nombre de colonnes à afficher.
- **`public String getColumnName(int col)`** : retourne le titre de la colonne **`col`** (commence à 0).

- **`public Object getValueAt(int row, int column)`** : retourne l'objet (String en général) à afficher en ligne **`row`**, colonne **`column`**.

L'exemple suivant hérite d'un TableModel généré. Il réduit le nombre de colonnes à afficher de 7 à 3. Les deux premières colonnes représentent deux Rubriques standard (numéro et nom du client). La troisième représente l'adresse du client, c'est-à-dire le résultat de la concaténation de la rue, du code postal et de la ville.

```
package test.swing;

import com.ibm.vap.generated.reuse.CustomerData;
public class NewCustomerTableModel extends
com.ibm.vap.generated.reuse.CustomerTableModel {

  public int getColumnCount (){
         return 3;
  }

  public String getColumnName (int i){
         if (i == 0) return "Id";
         if (i == 1) return "Name";
         if (i == 2) return "Address";
         return "";
  }

  public Object getValueAt(int row, int column){
         try {
                CustomerData data = (CustomerData) getRows().elementAt(row);
                if (column == 0) return data.getCusId();
                if (column == 1) return data.getCusNam();
                if (column == 2) {
                       String result = "" +
                              data.getStreet() +"." +
                              data getZipcod() + "-"+
                              data.getTown();
                       return result;
                }
         } catch (Throwable t) {}
         return null;
  }

}
```

5.1.3.9. Gestion de la présence des Rubriques

Les deux méthodes suivantes vous permettent de gérer la présence des Rubriques de la Vue Logique au niveau de la Proxy.

La méthode **`is<corub>Present`** permet de tester la présence ou l'absence de la Rubrique **`corub`**. Elle est générée pour toutes les classes **`DataDescription`** et **`UserDataDescription`**.

La méthode **`set<corub>Present(boolean aBoolean)`** permet d'indiquer la présence ou l'absence de la Rubrique avant toute méthode de mise à jour locale. Elle est générée pour toutes les classes **`DataDescription`** et **`UserDataDescription`**.

Par défaut, les Rubriques sont considérées comme étant absentes, sauf si une valeur par défaut a été indiquée dans la description VisualAge Pacbase.

5.1.3.10. Gestion du contrôle des Rubriques

Les trois méthodes suivantes vous permettent de gérer le contrôle des Rubriques de la Vue Logique au niveau de la Proxy.

La méthode **`get<corub>Index`** indique l'indice du champ **`corub`** dans la classe **`DataDescription`**. Cet indice est utilisé dans l'activation du contrôle serveur sur la Rubrique **`corub`**.

La méthode **`setcheck(int index, boolean aBoolean)`** permet d'activer ou d'inhiber les contrôles serveur sur une Rubrique (pointée par l'indice) avant toute mise à jour locale. Elle est générée pour toutes les classes **`DataDescription`** des nœuds racines ou dépendants dont le Composant Applicatif possède les options **`NULLMNGT=YES`** et **`CHECKSER=YES`** et un service de mise à jour.

Par défaut, toutes les Rubriques sont à contrôler (si l'attribut **`serverCheckOption`** est positionné à **`true`**).

5.1.3.11. Gestion des sous-schémas

Toute méthode serveur de sélection, lecture ou mise à jour prend en compte le sous-schéma présent dans la propriété **`subSchema`** et retourne donc les valeurs des Rubriques du sous-schéma. Si une sélection est suivie d'une pagination, le sous-schéma pris en compte est celui associé à la méthode de sélection.

Toute création locale est effectuée hors sous-schéma.

Toute modification/suppression locale est effectuée sur le sous-schéma associé à l'instance, c'est à dire :

- si la modification/suppression est effectuée sur une instance créée localement, le sous-schéma est vide.
- si la modification/suppression est effectuée sur une instance lue, le sous-schéma est celui associé à la sélection de cette instance.

De plus, les trois méthodes suivantes sont spécifiques à la gestion des sous-schémas.

La méthode **`resetSubSchema`** permet de réinitialiser la propriété **`subSchema`** à vide, c'est à dire de ne sélectionner aucun sous-schéma.

La méthode **`completeInstance`** permet de récupérer, par appel du Composant Applicatif associé à la Vue Logique, les valeurs des Rubriques n'appartenant pas au sous-schéma.

La méthode **`belongsToSubschema`** permet de savoir si la Rubrique passée en paramètre appartient au sous-schéma associé à l'instance contenue dans l'attribut **`detail`**.

5.1.4. Utilisation des événements

Les événements émis par une Proxy permettent de déclencher des méthodes applicatives appartenant à l'application graphique. Ces traitements sont exécutés en connectant un événement de la Proxy à une ou plusieurs méthodes de l'application graphique. L'exécution conditionnelle des méthodes est facilitée par le fait qu'un événement est toujours accompagné de son événement contraire, les deux événements ne pouvant jamais être émis simultanément.

La disponibilité d'un événement est fonction du type de Proxy. Tous les événements de l'interface publique sont documentés dans le *Manuel de Référence Clients Graphiques : Interface Publique des composants générés*.

5.1.4.1. Gestion événementielle des lectures massives

La gestion événementielle des lectures massives permet au développeur d'avoir des informations sur l'état de la collection d'instances contenu dans un noeud. Chaque méthode de pagination disponible propose son propre système de pagination événementielle.

La méthode de pagination en mode non-extend peut émettre les quatre événements suivants :

- `noPageBefore` : Cet événement est émis par un noeud racine ou référence à la fin de l'exécution d'une méthode de sélection de collection ou de pagination lorsque celle-ci ne renvoie pas d'erreur et que la page lue est la première de la collection courante.

- `pageBefore` : Cet événement est émis par un noeud racine ou référence à la fin de l'exécution d'une méthode de sélection de collection ou de pagination lorsque celle-ci ne renvoie pas d'erreur et que la page lue n'est pas la première de la collection courante.

- `noPageAfter` : Cet événement est émis par un noeud racine ou référence à la fin de l'exécution d'une méthode de sélection de collection ou de pagination lorsque celle-ci ne renvoie pas d'erreur et que la page lue est la dernière de la collection courante.

- `pageAfter` : Cet événement est émis par un noeud racine ou référence à la fin de l'exécution d'une méthode de sélection de collection ou de pagination lorsque celle-ci ne renvoie pas d'erreur et que la page lue n'est pas la dernière de la collection courante.

La méthode de pagination en mode extend peut émettre les deux événements suivants :

- `pageAfter` : Cet événement est émis par tout type de noeud à la fin de l'exécution d'une méthode de sélection de collection ou de pagination avant lorsque celle-ci ne renvoie pas d'erreur et que le nombre d'instances contenu dans le noeud ne correspond pas au nombre d'instances total contenu dans la base.

- `noPageAfter` : Cet événement est émis par tout type de noeud à la fin de l'exécution d'une méthode de sélection de collection ou de pagination avant lorsque celle-ci ne renvoie pas d'erreur et que le nombre d'instances contenu dans le noeud correspond au nombre d'instances total contenu dans la base au moment de la requête.

5.1.4.2. Gestion événementielle des lectures d'une instance

Une Vue Logique peut être mappée sur une ou plusieurs entités de stockage physique. Dans ce contexte, la gestion événementielle de la lecture d'une instance de Vue Logique peut émettre deux événements :

- **notFound** lorsque l'instance recherchée n'existe pas dans la base de données. Cet événement peut être émis lorsque la Vue Logique est mappée sur une ou plusieurs tables.

- **notComplete** lorsque l'instance recherchée est incomplète. C'est-à-dire qu'au moins une table a satisfait au critère de sélection et au moins une table n'a pas satisfait à ce critère. Cet événement ne peut être émis que dans un contexte de multi-mapping.

5.2. Exemple d'une applet

5.2.1. Introduction

Ce sous-chapitre présente une applet VisualAge Java, c'est-à-dire un programme destiné à être exécuté dans un browser.

Cette applet est développée successivement avec la version 1 de VisualAge Java puis avec la version 2 de VisualAge Java, ce qui permet d'illustrer les spécificités de développement de chaque version.

Les composants Proxy insérés dans l'applet ont été générés avec les options **Générer des Beans** et :

- **Utiliser les classes IBM Enterprise Access Builder** pour la version 1.
- **Utiliser Swing** pour la version 2.

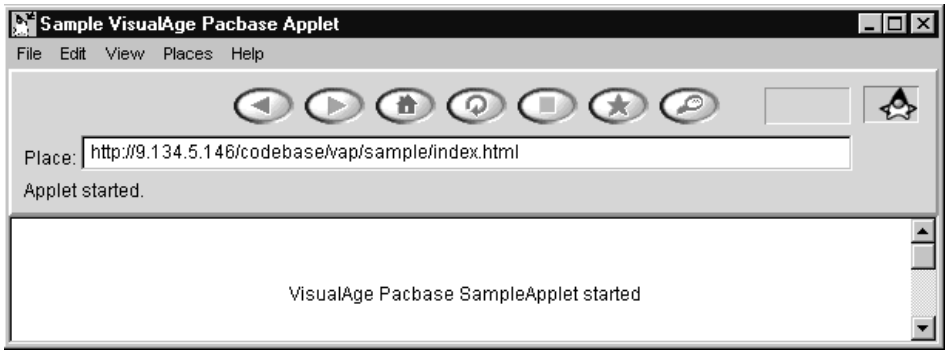
L'exemple manipule trois Proxy Elémentaires de la PVD :

- la Proxy Racine correspondant au noeud **Clients** qui gère les clients dans le système d'information décrit par le Dossier.
- la Proxy Dépendante correspondant au noeud **Commandes** qui gère les commandes dans le système d'information décrit par le Dossier.
- la Proxy Dépendante correspondant au noeud **Lignes de Commandes** qui gère les lignes de commandes dans le système d'information décrit par le Dossier.

5.2.2. Présentation de l'interface utilisateur

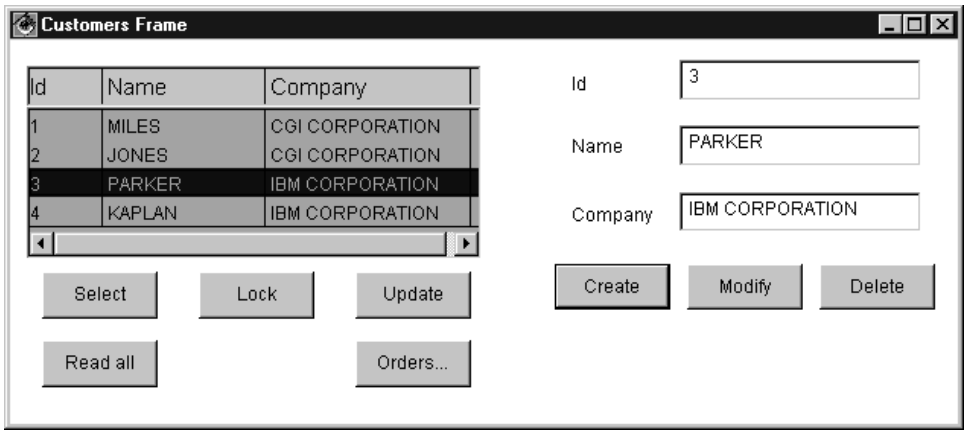
L'interface graphique utilisateur développée comprend l'applet elle-même et deux fenêtres.

- **L'applet**



L'applet n'a pas de fonctionnalité dans l'application utilisateur. Elle constitue un point de départ et permet à l'ensemble de l'application d'être accessible par le Web.

- **La fenêtre Customers**

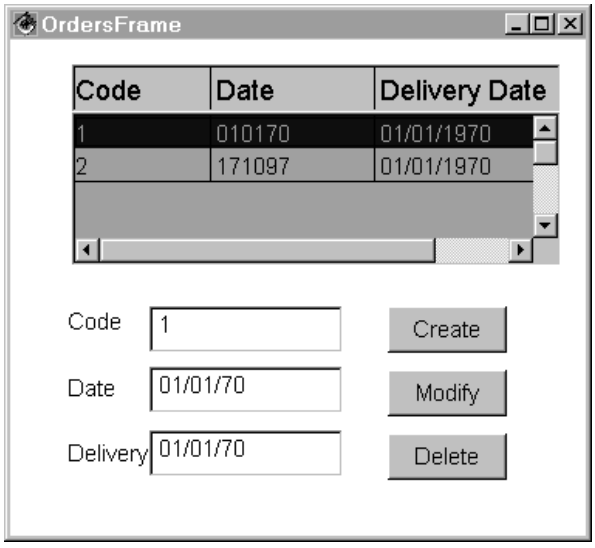


Cette fenêtre s'ouvre automatiquement lors du lancement de l'applet. Ses fonctionnalités sont les suivantes :

- Le bouton **`Select`** permet l'affichage de la liste des clients.
- Le bouton **`Read all`** permet, à partir de cette fenêtre, de demander la lecture des commandes du client sélectionné.
- Parce qu'un service de verrouillage a été spécifié dans le Dossier, l'utilisateur doit, avant toute mise à jour, cliquer sur le bouton **`Lock`** pour verrouiller l'instance sélectionnée dans la liste avant toute mise à jour.
- Le bouton **`Update`** permet de mettre à jour la base de données
- Le bouton **`Orders...`** permet la navigation vers la fenêtre de gestion des commandes.
- Le bouton **`Modify`** permet la modification d'un client à partir de la fiche, après verrouillage de cette instance.

  Après sélection d'un client dans la liste, l'utilisateur doit cliquer sur **`Lock`** pour s'approprier momentanément ce client. Il doit ensuite cliquer sur **`Modify`** après saisie des modifications souhaitées.

- **La fenêtre Orders**



La fenêtre **Orders** s'affiche lorsque l'utilisateur clique sur le bouton **Orders...** de la fenêtre **Customers**.

Pour le client sélectionné dans la fiche de la fenêtre **Customers**, cette fenêtre permet :

- de consulter la liste de ses commandes.
- de sélectionner une commande dans la liste et de l'afficher dans la fiche.
- de créer, modifier, supprimer des commandes.

5.2.3. Développement de l'interface utilisateur avec VisualAge Java V1

Le développement de l'interface graphique utilisateur comprend la programmation de l'applet et la construction des fenêtres **Customers** et **Orders**.

Dans la description de ces différentes étapes, l'utilisation des différents outils et fonctionnalités de VisualAge n'est pas détaillée. S'ils ne vous sont pas familiers, reportez-vous à la documentation appropriée.

Tout en gardant autant que possible une démarche séquentielle dans le développement, nous divisons, le cas échéant, la programmation des fenêtres en plusieurs parties selon des groupes cohérents de fonctionnalités.

A l'issue du traitement de chaque partie, nous présentons le Composition Editor tel qu'il doit apparaître en cachant éventuellement les liens précédemment développés pour mieux isoler ceux nouvellement créés.

5.2.3.1. Mise en place de l'exemple et création de l'applet

- Dans un projet, créez un package **vap.sample** destiné à contenir tous les composants de l'application.
- Dans ce package, créez une applet **SampleApplet**. Dans la fenêtre **SmartGuide – Create Applet**, indiquez l'option **Design the applet visually**, de manière à ce que le browser de classes s'ouvre directement sur l'onglet **Visual Composition**.

- **Affichage du texte**

Dans le Visual Composition Editor, effectuez les opérations suivantes :

- Redimensionnez l'applet.
- Dans l'applet, posez un bean **Label** (catégorie **Data Entry**). Dans la fenêtre **Properties** du bean, dans le champ **text**, entrez **VisualAge Pacbase SampleApplet started**.

- **Intégration de la Proxy Racine**

Pour poser la Proxy Racine sur la Free Form Surface, effectuez les opérations suivantes :

- Dans le menu **Options**, sélectionnez le choix **Add Bean**.
- Dans la fenêtre **Add Bean** qui s'ouvre alors, vous devez entrer **com.ibm.vap.generated.proxies.CustomerProxyLv** :
  - en le saisissant dans la zone correspondante,
  - ou en utilisant le bouton **Browse…**.

Le nom de classe, dans notre exemple **CustomerProxyLv**, doit être précédé du nom du package choisi lors de la génération, dans notre exemple, **com.ibm.vap.generated.proxies**.

- **Spécification de la communication avec la gateway**

A ce sujet, voir également le paragraphe **5.5.2.1**, *Accès direct au middleware ou mode local*.

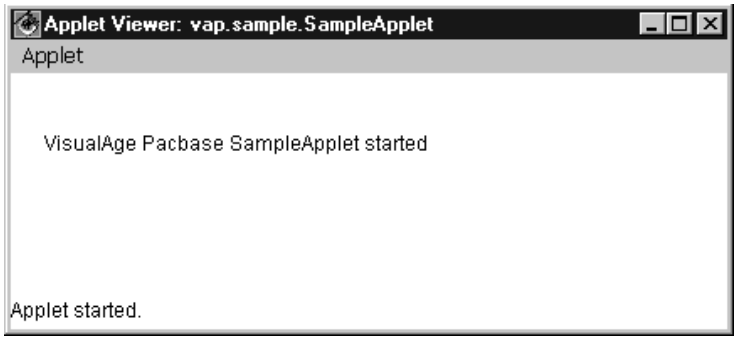
Dans cet exemple, on suppose que la Proxy Vue de Dossier communique avec son host, c'est-à-dire avec le serveur HTTP où l'applet est stockée. Pour que cette communication puisse s'effectuer, procédez de la manière suivante :

- Depuis un endroit quelconque de la Free Form Surface, accédez au menu contextuel.
- Sélectionnez **Tear-Off Property**, puis **codeBase (URL)**.
- Cliquez sur la Free Form Surface. Un bean variable **codeBase1** s'affiche alors.
- Accédez au menu contextuel du bean **codeBase1**, sélectionnez **Connect**, puis **All Features…..**
- Dans la fenêtre **Start connection from**, sélectionnez la propriété **host**. Un lien en pointillé s'affiche.
- Cliquez sur la Proxy Racine. Le menu contextuel s'affiche.
- Sélectionnez **All Features….** La fenêtre **Connect property named:** s'ouvre.
- Affichez toutes les propriétés disponibles pour la Proxy Racine en cochant **Show expert features**.
- Sélectionnez la propriété **Host**, puis cliquez sur **OK**.

La propriété **host** de **codeBase1** est maintenant connectée à la propriété **Host** de la Proxy Racine.

Ces connexions sont équivalentes à la ligne de code suivante dans l'applet : **getCustomerProxyLv1().setHost(getCodeBase().getHost());**

Vous pouvez maintenant tester votre applet. La fenêtre **Applet Viewer** s'affiche :



- **Programmation de l'ouverture de la fenêtre CustomersFrame depuis l'applet**

Cette étape se compose de deux parties, toutes deux à réaliser une fois les opérations décrites dans les paragraphe *Création de la fenêtre et intégration de la PVD dans le Composition Editor* et *Promotion de la Proxy Racine* effectuées :

- Appel de la Proxy Racine dans l'applet

- Dans le Composition Editor de l'applet, posez un bean constant de type **`CustomersFrame`** sur la Free Form Surface.
- Connectez la propriété **`this`** de la Proxy Racine à la propriété **`customerProxyLv1This`** du bean **`CustomersFrame`**.

- Programmation du lancement de la fenêtre depuis l'applet

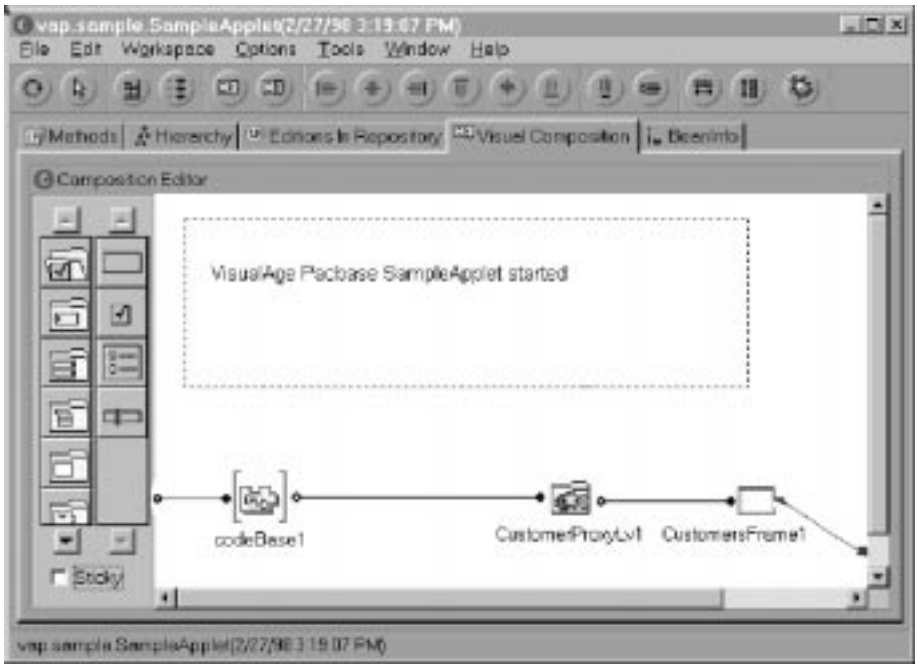
  Il s'agit maintenant de provoquer l'affichage automatique de la fenêtre C**`ustomersFrame`** immédiatement après le lancement de l'applet.

  Pour cela, connectez l'événement **`componentShown`** de l'applet à la méthode **`show`** du bean **`CustomersFrame`**.

- **Résultat dans le Composition Editor**

L'applet est maintenant terminée.

Voici le Composition Editor tel qu'il doit se présenter à ce stade :



5.2.3.2. Développement de la fenêtre Customers

5.2.3.2.1. Maquettage de rows et detail

Cette étape, après l'insertion de la Proxy Racine et sa promotion, détaille le maquettage de ses propriétés **`rows`** et **`detail`**.

- **Création de la fenêtre et intégration de la PVD dans le Composition Editor**

Créer une fenêtre de gestion des clients consiste à créer la classe correspondante dans le Workbench.

- Dans le Workbench, créez une classe **`CustomersFrame`** dans le package **`vap.sample`**.
- Dans la fenêtre **`SmartGuide –Create Class or Interface`** qui s'ouvre alors :

- entrez **java.awt.Frame** dans le champ **Superclass** : vous spécifiez ainsi que la classe **CustomersFrame** hérite de la classe **java.awt.Frame**.
- activez l'option **Design the class visually** de manière à ce que le browser de classes s'ouvre directement sur l'onglet **Visual Composition**.

- Pour intégrer la Proxy Vue de Dossier dans le Visual Composition Editor, effectuez les étapes décrites au paragraphe *Création de la fenêtre et intégration de la PVD dans le Composition Editor* à une différence près : dans la fenêtre **Add Bean**, sélectionnez l'option **Variable** dans la zone **Bean Type**.

  ☞ La Proxy Racine déposée ici est en effet de type variable car elle doit représenter la même instance de Proxy que celle appelée dans l'applet.

- **Promotion de la Proxy Racine**

L'objectif est maintenant d'assurer l'égalité permanente de cette Proxy variable avec la Proxy constante instanciée dans l'applet. Pour cela, il est nécessaire que la Proxy variable soit visible depuis l'extérieur de la classe **CustomersFrame**, pour qu'une connexion propriété - propriété entre la Proxy constante et la Proxy variable puisse être réalisée dans l'applet.

Effectuez les opérations suivantes :

- Depuis le menu contextuel de la Proxy, choisissez **Promote Bean feature**.
- Dans la colonne **Property** choisissez **this** puis cliquez sur **Promote**.

  La classe **CustomersFrame** a maintenant une propriété publique, en lecture/écriture, nommée **customerProxyLv1This** de type Proxy Racine.
- Sauvegardez la fenêtre (menu **File**, choix **Save Bean** ou **CTRL-F2**).

- **Maquettage de la propriété rows**

☞ Le maquettage proposé utilise un container EAB ; il n'est donc possible que si vous avez coché l'option **Utiliser les classes IBM Enterprise Access Builder** lors de la génération.

Cette étape inclut les opérations suivantes :

- A l'intérieur du bean **CustomersFrame**, posez un bean **Multi Column List Box**, container de la catégorie **Access Enterprise**. Redimensionnez ce bean.
- Vous devez maintenant maquetter dans le container les colonnes correspondant aux Rubriques associées au nœud racine.

  Ouvrez la fenêtre **Properties** du container. A partir du champ **columns** de cette fenêtre, ajoutez une colonne par Rubrique de la Vue Logique dans l'ordre dans lequel les Rubriques sont appelées dans le Référentiel. Vous pouvez maquetter autant de colonnes que la **DataDescription** de la Proxy a de propriétés ou ne maquetter que les premières.

  ☞ Les colonnes à maquetter correspondent aux valeurs retournées par la méthode **getAttributeStrings()** de la **DataDescription**.
- Une fois les colonnes créées, connectez la propriété **IRows** de la Proxy à la propriété **elements** du container.

☞ **IRows** est identique à **Rows** mais il renvoie une instance de la classe **IVector** qui, contrairement à la classe **DataDescriptionVector** retournée par **Rows**, est compatible avec la propriété **elements** du bean **Multi Column List Box**.

Cette opération équivaut réellement au maquettage de la propriété **Rows** de la Proxy.

- **Maquettage de la propriété detail**

Cette étape nécessite les opérations suivantes :

- Depuis la Proxy Racine, effectuez un Tear-Off de la propriété **detail**.
- Insérez un bean **label** par propriété à maquetter.
- Vous devez maintenant maquetter un champ de saisie par propriété à afficher.

  Pour cela, utilisez les beans **Pacbase Text Field**, **Pacbase Integer Field**, **Pacbase Decimal Field**, **Pacbase Date Field** et **Pacbase Time Field** fournis par Pacbench C/S lors de l'importation du runtime.

  Dans la palette, ces beans se trouvent dans la catégorie **VisualAge Pacbase**.

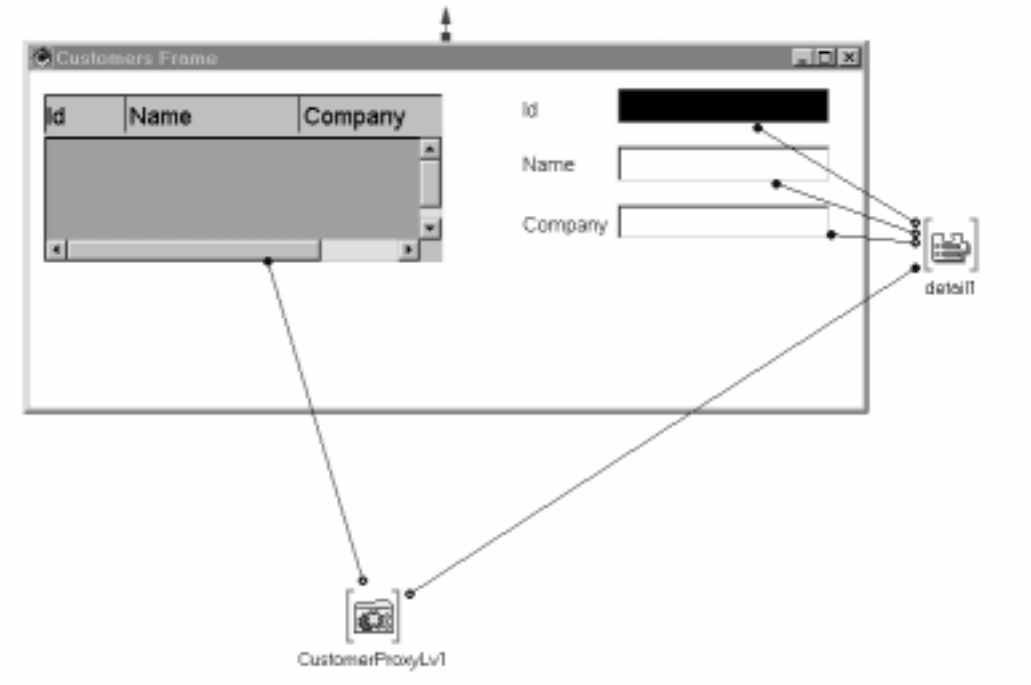
  Vous pouvez également insérer ces beans en sélectionnant dans le menu **Options** le choix **Add bean** ; entrez le nom du bean précédé du nom complet du package. Par exemple : **com.ibm.vap.beans.PacbaseDateField**.

  ☍ Pour des détails complémentaires sur ces beans, reportez-vous à la documentation en ligne des classes génériques.
- Insérez un bean de type approprié pour chaque propriété à maquetter.
- Ensuite, pour chaque champ maquetté, connectez la propriété du bean **detail** à sa propriété correspondante de type **String**, **Int**, **Decimal**, **Date** ou **Time**.

  Dans notre exemple, pour le champ de saisie correspondant au numéro du client, il faut connecter la propriété **Customer Number** du bean **detail** à la propriété **Int** du champ.

- **Résultat dans le Composition Editor**

A l'issue de ces étapes, le Composition Editor se présente comme ci-dessous :



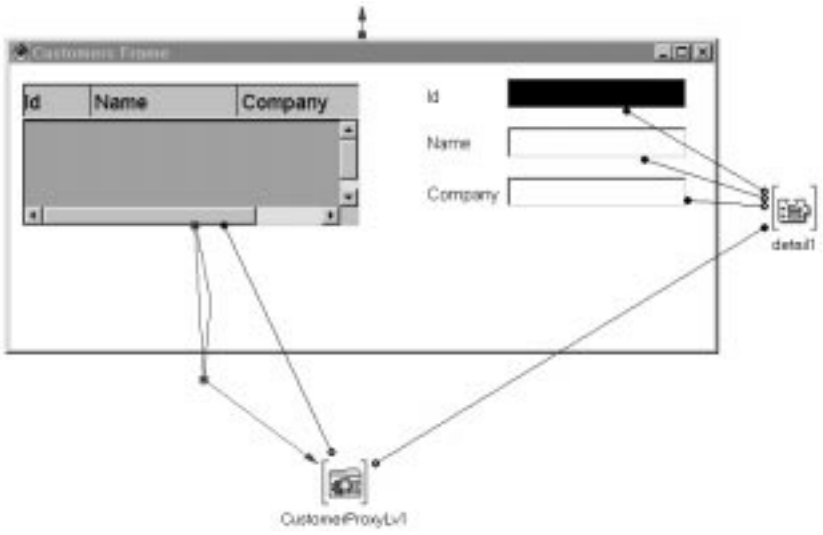
5.2.3.2.2. Sélection d'une instance dans rows et transfert dans detail

L'objectif est maintenant de programmer le transfert de l'instance sélectionnée par l'utilisateur depuis le container dans la propriété **`detail`** de la Proxy.

Pour cela, effectuez les étapes suivantes :

- Connectez l'événement **`itemStateChanged`** de la MultiColumnListBox à la méthode **`Get Detail From DataDescription`** de la Proxy Racine.
- Le lien apparaît en pointillés puisque la méthode **`Get Detail From DataDescription`** nécessite un paramètre.
- Pour spécifier ce paramètre, connectez la propriété **`selectedObject`** de la MultiColumnListBox au paramètre de type data du lien, ici **`customerData`**. Ce paramètre est accessible lorsque le pointeur se trouve au milieu de la connexion.

Voici le Composition Editor tel qu'il doit se présenter à l'issue de cette étape :



5.2.3.2.3. Déclenchement des méthodes de la Proxy et navigation vers la fenêtre Orders

- **Déclenchement des méthodes de la Proxy**

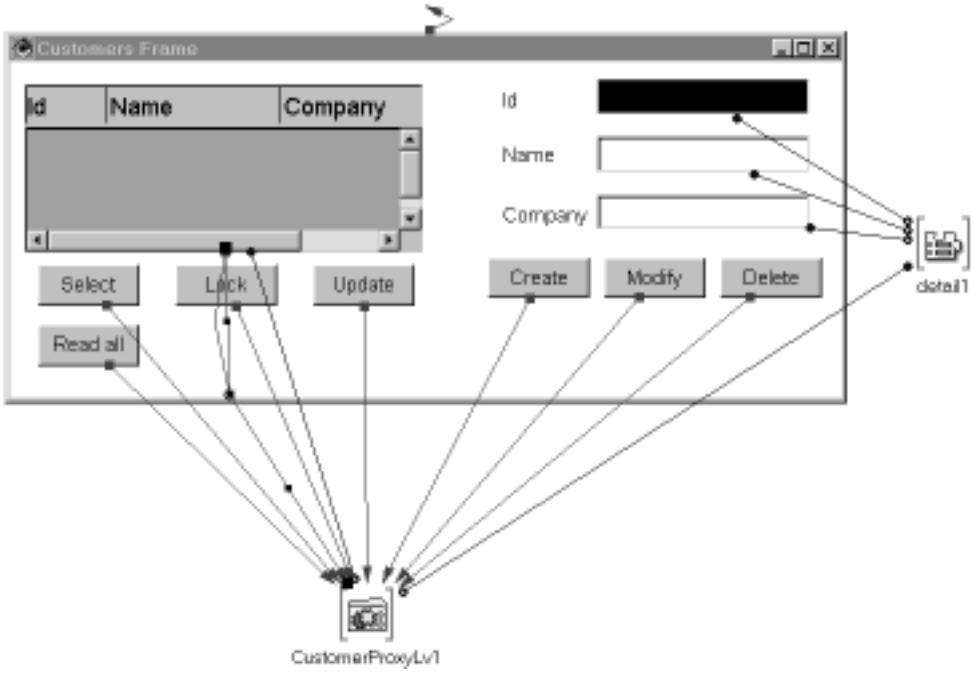
Pour chaque méthode que vous souhaitez programmer, posez un bouton-poussoir dans la fenêtre. Le libellé est modifiable dans la fenêtre **`Properties`** du bouton.

Pour déclencher une méthode, connectez l'événement **`actionPerformed`** d'un bouton donné à la méthode adéquate de la Proxy. Au besoin, si la méthode souhaitée n'est pas disponible, cochez l'option **`Show expert features`**.

Par exemple :

- pour programmer le déclenchement du bouton **`Update`**, il faut connecter le bouton, via l'événement **`actionPerformed`** à la Proxy, via la méthode **`Update Folder`**.
- pour programmer le déclenchement du bouton **`Read all`**, il faut connecter le bouton, via l'événement **`actionPerformed`** à la Proxy, via la méthode **`Read All Children From Detail`**.

A l'issue de ces opérations, le Composition Editor doit se présenter comme ci-dessous :

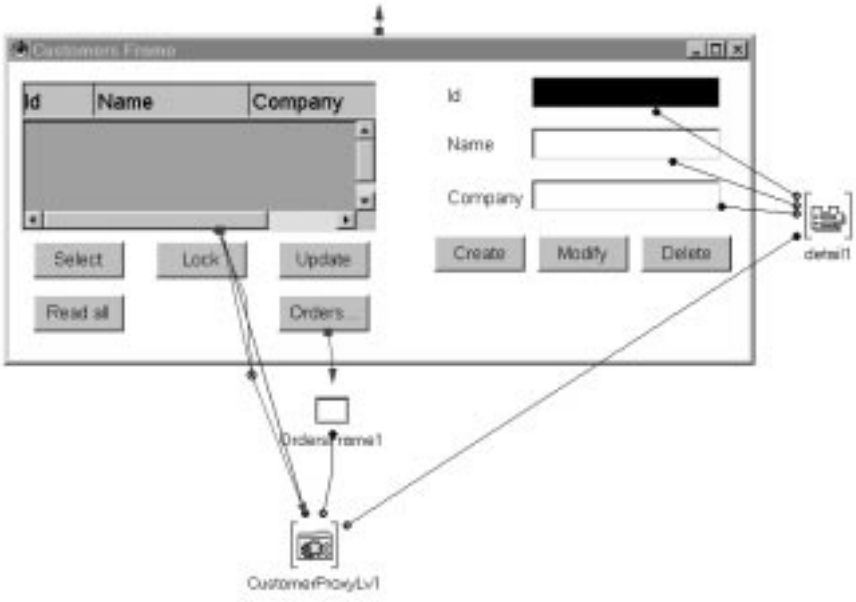


- **Gestion de la navigation**

Cette étape est à réaliser après la promotion de la Proxy Dépendante Orders.

- Sur la Free Form Surface, posez un bean constant **`OrdersFrame`**.
- Connectez la propriété **`Order Proxy`** de la Proxy Racine à la propriété **`orderProxyLv1This`** du bean **`OrdersFrame`**. Cette connexion assure le lien des deux objets Proxy entre les deux fenêtres.
- Posez un bouton **`Orders...`** dans le bean **`CustomersFrame`**.
- Connectez l'événement **`actionPerformed`** du bouton à la méthode **`show`** du bean **`OrdersFrame`**.

Voici le Composition Editor tel qu'il doit se présenter à l'issue de cette étape :



Pour une meilleure lisibilité, les liens entre les autres boutons et la Proxy Racine sont cachés.

5.2.3.3. Développement de la fenêtre Orders

Cette fenêtre présente les informations relatives aux commandes. Elle s'affiche lorsque l'utilisateur clique sur le bouton **`Orders…`** depuis la fenêtre **`Customers`**.

Le container affiche automatiquement les commandes du client sélectionné dans la fenêtre **`Customers`** si l'utilisateur clique sur **`Read all`** avant de cliquer sur **`Orders…`**.

Le développement de cette fenêtre qui manipule la Proxy Dépendante **`OrderProxyLv`** comporte les étapes décrites ci-dessous.

5.2.3.3.1. Création de la fenêtre Orders

Dans le Workbench, de la même manière que pour la création de la fenêtre **`Customers`**, créez une une classe **`OrdersFrame`** héritant de **`java.awt.Frame`** dans le package **`vap.sample`**.

☞ Pour plus de détails, reportez-vous au paragraphe **5.2.3.2**, point *Création de la fenêtre et intégration de la PVD dans le Composition Editor*.

5.2.3.3.2. Intégration et promotion de la Proxy Dépendante

Dans l'onglet Visual Composition, posez un bean variable **`OrderProxyLv`** sur la Free Form Surface.

☞ Pour plus de détails sur l'intégration de la Proxy, reportez-vous au paragraphe **5.2.3.2**, point *Création de la fenêtre et intégration de la PVD dans le Composition Editor*.

Il faut maintenant promouvoir cette Proxy variable de manière à la rendre visible de l'extérieur.

Pour cela, effectuez les opérations suivantes :

- Depuis le menu contextuel de la Proxy, choisissez **Promote Bean feature**.
- Dans la colonne **Property** choisissez **this** puis cliquez sur **Promote**.

  La classe **OrdersFrame** a maintenant une propriété publique, en lecture/écriture, nommée **OrderProxyLv1This** de type Proxy Dépendante.
- Sauvegardez la fenêtre (menu **File**, choix **Save Bean** ou **CTRL-F2**).

5.2.3.3.3. Maquettage de rows et detail de la Proxy Dépendante

Les opérations à réaliser ici sont identiques à celles requises pour le maquettage des propriétés **rows** et **detail** de la Proxy Racine.

Le même container EAB est utilisé pour le maquettage de **rows**.

Pour plus de détails, reportez-vous au point **5.2.3.2.1**, *Maquettage de rows et detail*.

5.2.3.3.4. Sélection d'une instance dans rows et transfert dans detail

Les opérations à effectuer ici sont identiques à celles nécessaires pour les propriétés **rows** et **detail** de la Proxy Racine.

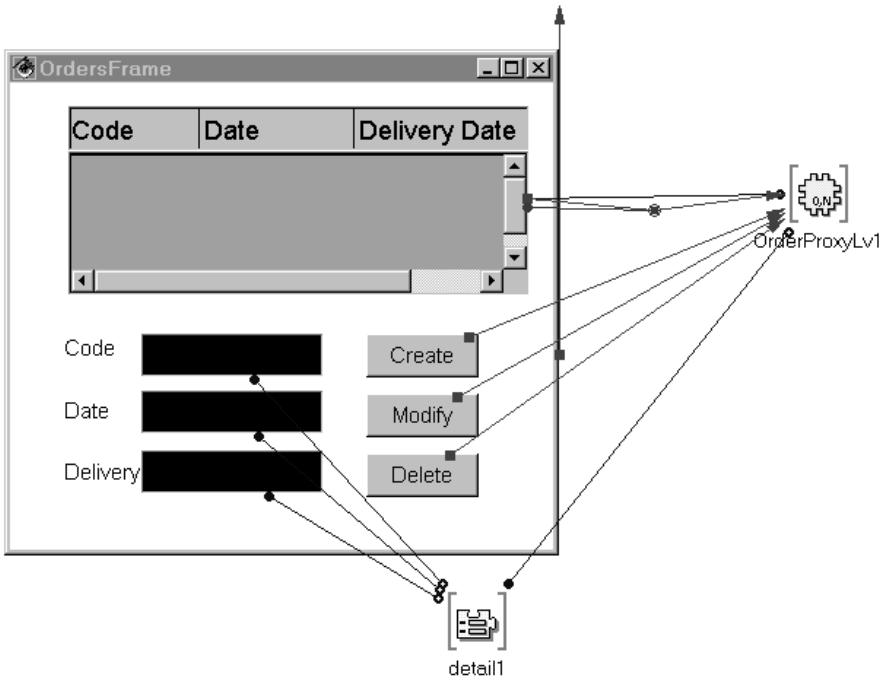
Pour plus de détails, reportez-vous au point **5.2.3.2.2**.

5.2.3.3.5. Déclenchement des méthodes de la Proxy Dépendante

Utilisez les mêmes principes que ceux décrits dans le point **5.2.3.2.3**.

5.2.3.3.6. Résultat dans le Composition Editor

Une fois toutes ces opérations effectuées, le Composition Editor se présente comme ci-dessous :



5.2.4. Développement de l'interface utilisateur avec VisualAge Java V2

Cette section détaille le développement, avec VisualAge Java V2, de l'application dont l'interface utilisateur est présentée à la section 5.2.2 *Présentation de l'interface utilisateur*.

Les objets Proxy intégrés dans l'application ont été générés avec les options **`Generate Beans`** et **`Utiliser Swing`**. Tous les composants insérés sont des beans Swing.

Nous partons du principe que nous créons l'application depuis le début. Nous ne reprenons donc pas l'application développée avec VisualAge Java V1.

☞ Si vous avez déjà développé une application avec VisualAge Java V1, vous pouvez la sauvegarder et continuer à la développer avec VisualAge Java V2. Cependant, si vous voulez insérer des composants Swing, vous devez regénérer les objets Proxy avec l'option **`Utiliser Swing`** afin que ces objets Proxy communiquent correctement avec les beans swing.

VisualAge Java V2 ne reconnaît pas les containers EAB, qui sont des composants spécifiques de VisualAge Java V1. Si l'application développée avec la V1 inclut de tels composants, vous devez les remplacer par d'autres composants (composant swing JTable par exemple).

5.2.4.1. Mise en place de l'exemple et création de l'applet

- Dans un projet, créez un package **`example.swing`** destiné à contenir tous les composants de l'application.
- Dans ce package, créez une applet **`SwingApplet`**. Dans la fenêtre **`SmartGuide – Create Applet`**, sélectionnez, via le bouton **`Browse`**, **`JApplet`** dans la zone **`SuperClass`** et cochez l'option **`Compose the applet visually`**, de manière à ce que le browser de classes s'ouvre directement sur l'onglet **`Visual Composition`**.

- **Affichage du texte**

Dans le Visual Composition Editor, effectuez les opérations suivantes :

- Redimensionnez l'applet.
- Dans l'applet, posez un bean **`JLabel`**. Dans la fenêtre **`Properties`** du bean, dans le champ **`text`**, entrez **`VisualAge Pacbase SwingSampleApplet started`**.

- **Intégration de la Proxy Racine**

Pour poser la Proxy Racine sur la Free Form Surface, effectuez les opérations suivantes :

- Cliquez sur l'icône **`Choose Bean`** située au-dessus de la palette.
- Sélectionnez le type de bean **`Class`**.
- Avec le bouton Browse, sélectionnez le nom de la classe **`CustomerProxyLv`**.

- **Spécification de la communication avec la gateway**

A ce sujet, voir également le paragraphe **5.5.2.1**, *Accès direct au middleware ou mode local*.

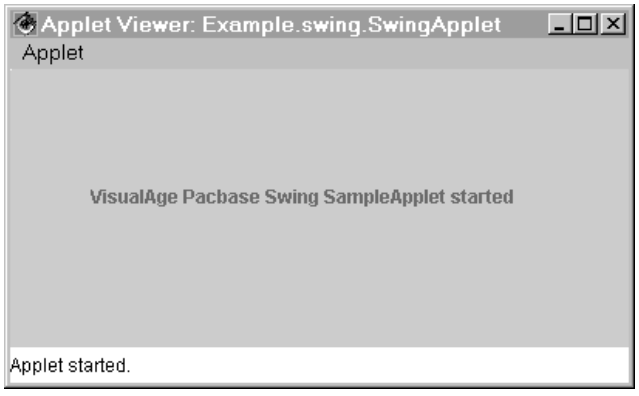
Dans cet exemple, on suppose que la Proxy Vue de Dossier communique avec son host, c'est-à-dire avec le serveur HTTP où l'applet est stockée. Pour que cette communication puisse s'effectuer, procédez de la manière suivante :

- Depuis un endroit quelconque de la Free Form Surface, accédez au menu contextuel.
- Sélectionnez **Tear-Off Property**, puis **codeBase (URL)**.
- Cliquez sur la Free Form Surface. Un bean variable **codeBase1** s'affiche alors.
- Accédez au menu contextuel du bean **codeBase1**, sélectionnez **Connect**, puis **Connectable Features…**.
- Dans la fenêtre **Start connection from**, sélectionnez la propriété **host**. Un lien en pointillé s'affiche.
- Cliquez sur la Proxy Racine. Le menu contextuel s'affiche.
- Sélectionnez **Connectable Features…**. La fenêtre **End Connection to** s'ouvre.
- Affichez toutes les propriétés disponibles pour la Proxy Racine en cochant **Show expert features**.
- Sélectionnez la propriété **Host**, puis cliquez sur **OK**.

La propriété **host** de **codeBase1** est maintenant connectée à la propriété **Host** de la Proxy Racine.

Ces connexions sont équivalentes à la ligne de code suivante dans l'applet : **getCustomerProxyLv1().setHost(getCodeBase().getHost());**

Vous pouvez maintenant tester votre applet. La fenêtre **Applet Viewer** s'affiche :



- **Programmation de l'ouverture de la fenêtre CustomersFrame depuis l'applet**

Cette étape se compose de deux parties, toutes deux à réaliser une fois les opérations décrites dans les paragraphe *Création de la fenêtre et intégration de la PVD dans le Composition Editor* et *Promotion de la Proxy Racine* effectuées :

- Appel de la Proxy Racine dans l'applet
  - Dans le Composition Editor de l'applet, posez un bean constant de type **CustomersFrame** sur la Free Form Surface.

Pour cela : sélectionnez l'icône **Choose Bean** au-dessus de la palette, choisissez le type de bean **Class** et sélectionnez, avec le bouton Browse, la classe **CustomersFrame**.

- Connectez la propriété **this** de la Proxy Racine à la propriété **customerProxyLv1This** du bean **CustomersFrame**.

- Programmation du lancement de la fenêtre depuis l'applet

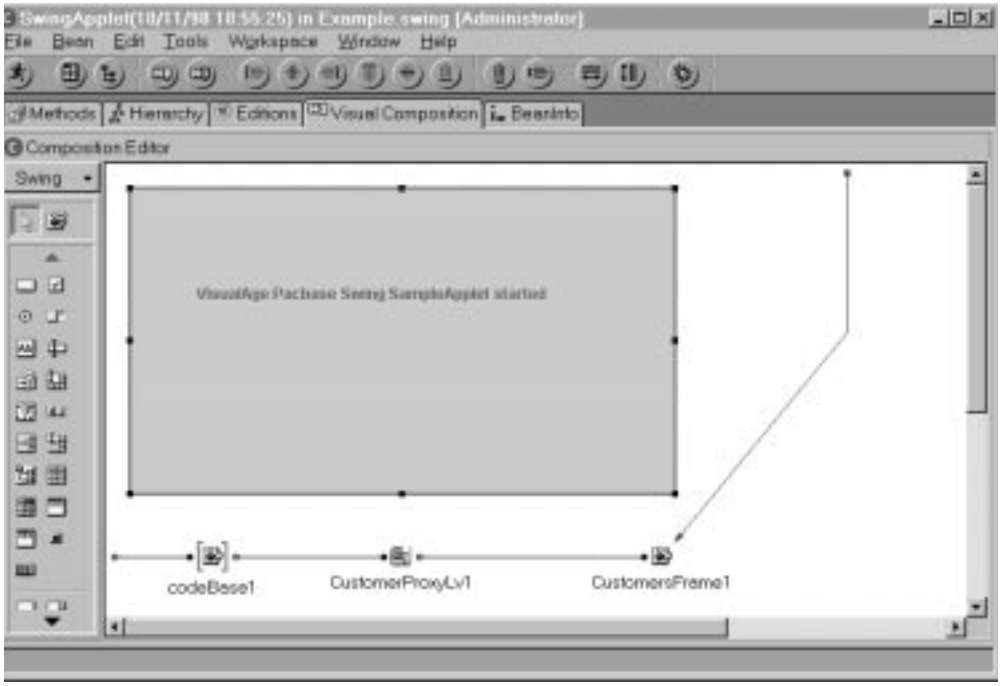
  Il s'agit maintenant de provoquer l'affichage automatique de la fenêtre **CustomersFrame** immédiatement après le lancement de l'applet.

  Pour cela, connectez l'événement **componentShown** de l'applet à la méthode **show** du bean **CustomersFrame**.

- **Résultat dans le Composition Editor**

L'applet est maintenant terminée.

Voici le Composition Editor tel qu'il doit se présenter à ce stade :



5.2.4.2. Développement de la fenêtre Customers

5.2.4.2.1. Maquettage de rows et detail

Cette étape, après l'insertion de la Proxy Racine et sa promotion, détaille le maquettage de ses propriétés **rows** et **detail**.

- **Création de la fenêtre et intégration de la PVD dans le Composition Editor**

Créer une fenêtre de gestion des clients consiste à créer la classe correspondante dans le Workbench.

- Dans le Workbench, créez une classe **CustomersFrame** dans le package **example.swing**.
- Dans la fenêtre **SmartGuide –Create Class or Interface** qui s'ouvre alors :

- sélectionnez **JFrame** avec le bouton **Browse** dans le champ **Superclass** : vous spécifiez ainsi que la classe **CustomersFrame** hérite de la classe **com.sun.java.swing.JFrame**.
- activez l'option **Compose the class visually** de manière à ce que le browser de classes s'ouvre directement sur l'onglet **Visual Composition**.

- Pour intégrer la Proxy Vue de Dossier dans le Visual Composition Editor :
- Cliquez sur l'icône **Choose Bean** situé au-dessus de la palette.
- Sélectionnez le type de bean **Variable**.
- Avec le bouton Browse, sélectionnez le nom de la classe **CustomerProxyLv**.

  ☞ La Proxy Racine déposée ici est en effet de type variable car elle doit représenter la même instance de Proxy que celle appelée dans l'applet.

- **Promotion de la Proxy Racine**

L'objectif est maintenant d'assurer l'égalité permanente de cette Proxy variable avec la Proxy constante instanciée dans l'applet. Pour cela, il est nécessaire que la Proxy variable soit visible depuis l'extérieur de la classe **CustomersFrame**, pour qu'une connexion propriété - propriété entre la Proxy constante et la Proxy variable puisse être réalisée dans l'applet.

Effectuez les opérations suivantes :

- Depuis le menu contextuel de la Proxy, choisissez **Promote Bean feature**.
- Dans la colonne **Property** choisissez **this** puis cliquez sur **>>**.

  La classe **CustomersFrame** a maintenant une propriété publique, en lecture/écriture, nommée **customerProxyLv1This** de type Proxy Racine.
- Sauvegardez la fenêtre (menu **Bean**, choix **Save Bean**).

- **Maquettage de la propriété rows**

A l'intérieur du bean **CustomersFrame**, posez un bean **JTable** et redimensionnez-le.

Pour voir apparaître les colonnes de la JTable, vous devez exécuter la JFrame. Les colonnes de la JTable sont initialisées avec les Rubriques de la Vue Logique et les lignes représentent les instances présentes dans **Rows**. Le contenu de la JTable est rafraîchi après chaque mise à jour de **Rows** (sélections, créations…).

Deux maquettages sont possibles :

- Si vous voulez afficher toutes les colonnes correspondant à toutes les Rubriques de la Vue Logique avec les noms en clair définis dans la Proxy, il vous suffit de connecter la propriété **Table model** de la Proxy à la propriété **model** de la JTable.
- En revanche, si vous voulez sélectionner les colonnes à afficher, modifier leur en-tête, ou créer une nouvelle colonne affichant des informations calculées en local, vous devez personnaliser la JTable.

  Pour cela, vous devez créer une nouvelle classe **TableModel** et personnaliser ses méthodes.

Cette nouvelle classe doit être publique et doit hériter:

- soit de **CustomerTableModel**, situé dans le package **com.ibm.vap.generated.reuse**. Ainsi, cette classe héritera automatiquement de toutes les méthodes existant dans **CustomerTableModel** et vous ne modifierez que les méthodes qui ne conviennent pas.
- soit directement de **PacbaseTableModel**, situé dans le package **com.imb.vap.beans.swing**. Dans ce cas, vous devrez réécrire toutes les méthodes que vous voulez utiliser puisqu'elles ne sont pas récupérées automatiquement.

Pour créer la nouvelle classe, sélectionnez le choix **Add Class** dans le menu contextuel du package (**com.ibm.vap.generated.reuse** ou **com.ibm.vap.beans.swing**). Donnez-lui un nom (par exemple, **NewCustomerTableModel**) et dans le champ **Superclass**, sélectionnez la classe (**CustomerTableModel** ou **PacbaseTableModel**) dont cette nouvelle classe va hériter.

Vous devez ensuite personnaliser les méthodes de cette classe en insérant du code directement dans la partie **source**. Dans notre exemple, nous avons choisi de faire hériter la nouvelle classe **NewCustomerTableModel** de la classe **CustomerTableModel**.

Nous devons personnaliser la méthode **public int getColumnCount()** pour limiter le nombre de colonnes de la JTable à 3, alors que la Vue Logique contient 7 Rubriques.

```
public class NewCustomerTableModel extends
com.ibm.vap.generated.reuse.CustomerTableModel {
  public int getColumnCount (){
         return 3;
  }
}
```

Les autres méthodes permettant de personnaliser les colonnes de la JTable sont documentées au point **5.1.3.8**.

Dans le Composition Editor, il vous suffit alors de :

- Poser une instance du nouveau **TableModel**,
- Connecter cette instance, via sa propriété **this**, à la propriété **TableModel** de la Proxy,
- Connecter cette instance, via sa propriété **this**, à la propriété **model** de la JTable.

- **Maquettage de la propriété detail**

Cette étape nécessite les opérations suivantes :

- Depuis la Proxy Racine, effectuez un Tear-Off de la propriété **detail**.
- Insérez un bean **JLabel** par propriété à maquetter.
- Vous devez maintenant maquetter un champ de saisie par propriété à afficher.

  Pour cela, utilisez les beans **Pacbase Swing Text Field** et **Pacbase Swing Integer Field** fournis par Pacbench C/S lors de l'importation du runtime.

Vous pouvez également insérer ces beans en cliquant sur l'icône **Choose Beans** située en haut de la palette. Sélectionnez le type de bean **Class** ou **Variable**. Puis sélectionnez, à l'aide du bouton Browse, le nom du bean : par exemple : **PacbaseJTextField**.

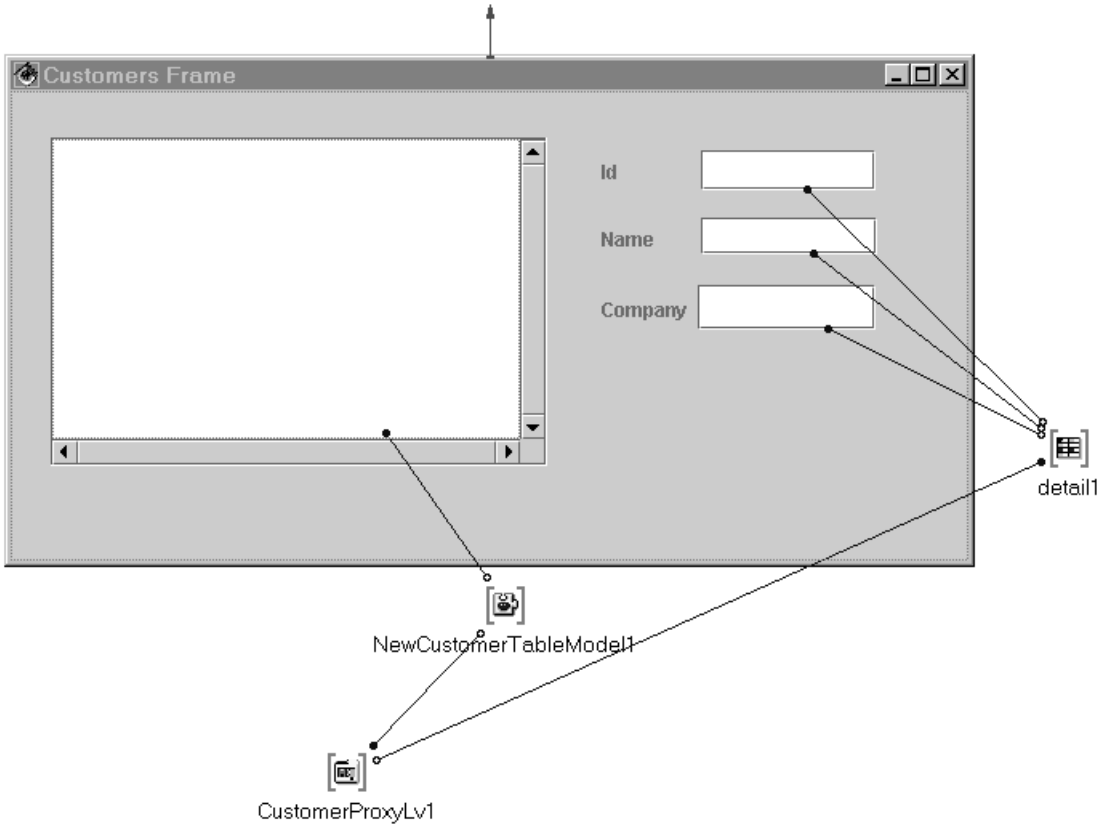
Pour des détails complémentaires sur ces beans, reportez-vous à la documentation en ligne des classes génériques.

- Insérez un bean de type approprié pour chaque propriété à maquetter.
- Ensuite, pour chaque champ maquetté, connectez la propriété du bean **detail** à la propriété correspondante de type **String** ou **Int**.

  Dans notre exemple, pour le champ de saisie correspondant au numéro du client, il faut connecter la propriété **Customer Number** du bean **detail** à la propriété **Int** du champ.

- **Résultat dans le Composition Editor**

A l'issue de ces étapes, le Composition Editor se présente comme ci-dessous :



Nous avons choisi ici de personnaliser la JTable. C'est pourquoi nous avons inséré un bean NewCustomerTableModel1.

5.2.4.2.2. Sélection d'une instance dans rows et transfert dans detail

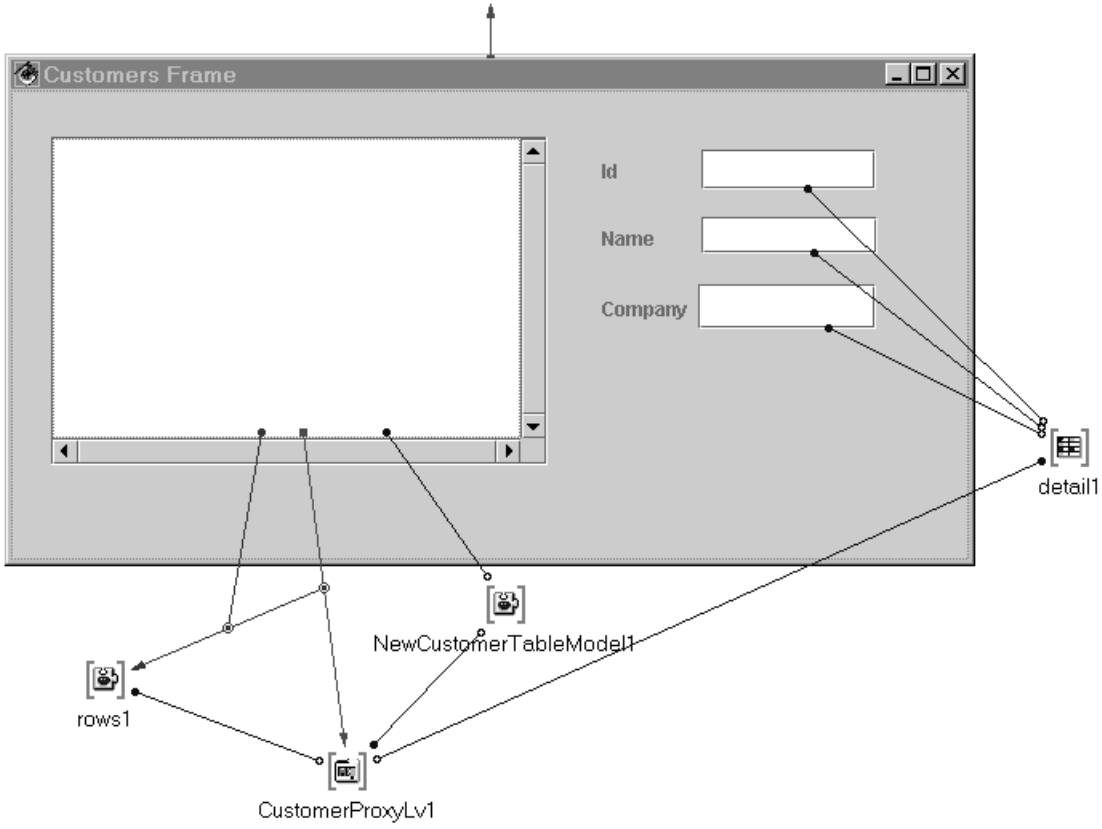
L'objectif est maintenant de programmer le transfert de l'instance sélectionnée par l'utilisateur depuis la table dans la propriété **detail** de la Proxy.

Pour cela, effectuez les étapes suivantes :

- Connectez les événements **keyEvents** et **mouseEvents** de la JTable à la méthode **Get Detail From DataDescription** de la Proxy Racine.

- Les liens apparaissent en pointillés puisque la méthode **`Get Detail From DataDescription`** nécessite un paramètre.
- Faites un **`Tear-Off`** de la propriété **`Rows`** de la Proxy Racine.Vous obtenez alors un bean variable **`rows1`**.
- Pour spécifier le paramètre requis par la méthode **`Get Detail From DataDescription`**, cliquez successivement sur les liens en pointillés. Connectez le paramètre de type data du lien, ici **`customerData`**, à la méthode **`elementAt(int)`** de **`rows1`**.
- Connectez ensuite la propriété **`SelectedRow`** de la JTable au paramètre **`Arg1`** du lien créé lors de l'étape précédente. Ce paramètre est accessible lorsque le pointeur se trouve au milieu de la connexion.

Voici le Composition Editor tel qu'il doit se présenter à l'issue de cette étape :



Pour plus de clarté, nous ne présentons la connexion que d'un des deux événements de la JTable.

5.2.4.2.3. Déclenchement des méthodes de la Proxy et navigation vers la fenêtre Orders

- **Déclenchement des méthodes de la Proxy**

Le déclenchement des méthodes est identique à celui détaillé pour la Proxy Racine de l'exemple VisualAge Java V1.

Pour plus de détails, reportez-vous au point **5.2.3.2.3**.

- **Gestion de la navigation**

La gestion de la navigation est identique à celle détaillée pour la Proxy Racine de l'exemple VisualAge Java V1.

☞ Pour plus de détails, reportez-vous au point **5.2.3.2.3**.

5.2.4.3. Développement de la fenêtre Orders

La fenêtre Orders est la même que celle développée dans l'exemple VisualAge Java V1.

☞ Pour plus de détails, reportez-vous au paragraphe **5.2.3.3.**

5.2.4.3.1. Création de la fenêtre Orders

Dans le Workbench, de la même manière que pour la création de la fenêtre **`Customers`**, créez une classe **`OrdersFrame`** héritant de **`JFrame`** dans le package **`example.swing`**.

☞ Pour plus de détails, reportez-vous au paragraphe **5.2.4.2**, point *Création de la fenêtre et intégration de la PVD dans le Composition Editor*.

5.2.4.3.2. Intégration et promotion de la Proxy Dépendante

Dans l'onglet Visual Composition, posez un bean variable **`OrderProxyLv`** sur la Free Form Surface.

☞ Pour plus de détails sur l'intégration de la Proxy, reportez-vous au paragraphe **5.2.4.2**, point *Création de la fenêtre et intégration de la PVD dans le Composition Editor*.

Il faut maintenant promouvoir cette Proxy variable de manière à la rendre visible de l'extérieur.

Pour cela, effectuez les opérations suivantes :

Effectuez les opérations suivantes :

- Depuis le menu contextuel de la Proxy, choisissez **`Promote Bean feature`**.
- Dans la colonne **`Property`** choisissez **`this`** puis cliquez sur **`>>`**.

  La classe **`OrdersFrame`** a maintenant une propriété publique, en lecture/écriture, nommée **`OrderProxyLv1This`** de type Proxy Dépendante.
- Sauvegardez la fenêtre (menu **`Bean`**, choix **`Save Bean`**).

5.2.4.3.3. Maquettage de rows et detail de la Proxy Dépendante

Le maquettage est identique à celui détaillé pour la Proxy Racine.

☞ Pour plus de détails, reportez-vous au point **5.2.4.2.1**, *Maquettage de rows et detail*.

5.2.4.3.4. Sélection d'une instance dans rows et transfert dans detail

Les opérations à effectuer ici sont identiques à celles nécessaires pour les propriétés **`rows`** et **`detail`** de la Proxy Racine.

☞ Pour plus de détails, reportez-vous au point **5.2.4.2.2**.

5.2.4.3.5. Déclenchement des méthodes de la Proxy Dépendante

Utilisez les mêmes principes que ceux décrits dans le point **5.2.3.2.3**.

5.3. Particularités du développement d'une application standalone

5.3.1. Introduction

Ce sous-chapitre a pour objectif de souligner les différences entre le développement d'une applet et celui d'une application *standalone*.

☞ Une application *standalone*, contrairement à une applet, n'est pas destinée à être exécutée dans un browser Web.

Les différences fondamentales sont les suivantes :

- Dans le développement d'une application *standalone*, la classe de base utilisée est de type **`Frame`** (pour awt) ou **`JFrame`** (pour swing). Cette classe hérite de la classe **`java.awt.Frame`** (pour awt) ou **`com.sun.java.swing.JFrame`** (pour swing) alors qu'une applet hérite de la classe **`java.applet.Applet`** (pour awt) ou **`java.applet.JApplet`** (pour swing).
- Dans le développement d'une application *standalone*, l'instanciation de la Proxy Racine est conditionnée par l'utilisation de la méthode **`setLocationsFile`** et non par celle des propriétés **`Host`** et **`Port`**. Cette méthode permet de positionner le fichier des *locations* **`VAPLOCAT.INI`** pour que l'application accède directement au middleware et non via la gateway.

5.3.2. Exemple

Cet exemple montre comment modifier l'applet précédemment développée en application *standalone*. Il réutilise les fenêtres **`CustomersFrame`**, désormais le point d'entrée de votre application, et **`OrdersFrame`**.

☞ Pour les fenêtres développées avec l'option Swing de VisualAge Java V2, il suffit de remplacer les classes awt présentes dans le code par les classes correspondantes swing.

Aucune modification n'est nécessaire dans le Composition Editor.

Il vous suffit d'insérer du code spécifique dans la méthode **`main`** de la classe **`CustomersFrame`**. La méthode **`main`** constitue le point d'entrée d'une application *standalone*.

Ce code spécifique vous est présenté ci-dessous entre les lignes de commentaires ***`//begin`*** et ***`//end`***.

```
/**
 * main entrypoint - starts the part when it is run as an application
 * @param args java.lang.String[]
 */
public static void main(java.lang.String[] args) {
   try {
      vap.sample.CustomersFrame aCustomersFrame = new vap.sample.CustomersFrame();
      try {
         Class aCloserClass = Class.forName(« uvm.abt.edit.WindowCloser »);
         Class parmTypes[] = { java.awt.Window.class };
         Object parms[] = { aCustomersFrame };
         java.lang.reflect.Constructor aCtor = aCloserClass.getConstructor(parmTypes);
```

```
      aCtor.newInstance(parms);
} catch (java.lang.Throwable exc) {};
//begin
aCustomersFrame.setCustomerProxyLv1(new
com.ibm.vap.generated.proxies.CustomerProxyLv());
aCustomersFrame.getCustomerProxyLv1().setLocationsFile(« d:\\user\\vapb\\vaplocat.ini
 »);
//end
aCustomersFrame.setVisible(true);
catch (Throwable exception) {
  System.err.println(« Exception occurred in main() of java.awt.Frame »);
}
}
```

La première instruction spécifie la création d'une nouvelle Proxy Racine. La précédente est déjà instanciée dans l'applet et ne peut être réutilisée ici comme bean constant.

La seconde instruction positionne le fichier des *locations*.

5.4. Gestion des erreurs

Il n'y a pas de message d'erreur lors des phases d'extraction et de génération.

Il est donc indispensable d'avoir compilé correctement le Composant Applicatif que l'on veut extraire car

- la commande GVC de la procédure GPRT extrait toujours sans afficher d'erreur, même si le Composant Applicatif ne contient aucune Vue Logique,
- le générateur ne fait aucun contrôle sur le fichier qu'il prend en entrée.

  En revanche, des messages d'erreur peuvent apparaître lors des phases de développement, de test et d'exécution de l'application. Ces messages proviennent d'erreurs locales, serveur ou de communication.

5.4.1. Principes

5.4.1.1. Introduction

La gestion des erreurs associées à la manipulation des objets Proxy VisualAge Pacbase se base sur le mécanisme des levées d'exceptions propres au langage Java.

Les objets Proxy peuvent lever quatre types d'erreurs Pacbench C/S ainsi que l'ensemble des erreurs Java.

Pour la gestion des erreurs Pacbench C/S, l'utilisateur dispose de quatre classes génériques dont les méthodes permettent de véhiculer les erreurs levées par les objets Proxy. Chacune de ces classes correspond à un type d'erreurs :

- Erreurs locales
- Erreurs serveur
- Erreurs système
- et erreurs de communication

Ces classes héritent toutes de la classe **`java.lang.Throwable`** et se trouvent dans le package **`com.ibm.vap.generic`**.

5.4.1.2. Programmation

Les exceptions doivent obligatoirement être interceptées par la programmation dans le composant Client.

La programmation des erreurs nécessite d'une part l'écriture de code Java, et d'autre part la construction graphique de la fenêtre qui permet d'afficher les erreurs.

☞ Un exemple de gestion des erreurs vous est proposé en section 5.4.6.

☞ Pour plus de détails sur les exceptions susceptibles d'être levées par les objets Proxy, consultez le *Manuel de Référence Clients Graphiques : Interface Publique des composants générés* ou, dans votre station VisualAge, reportez-vous à la signature des méthodes correspondantes.

5.4.2. Erreurs locales

Les erreurs locales provoquent l'exception **`com.ibm.vap.generic.LocalException`**. Cette exception porte une propriété de type **`int`** permettant d'identifier le type de l'erreur.

☞ Les erreurs à l'origine de cette exception sont également décrites dans la documentation HTML associée aux classes génériques : Package **`com.ibm.vap.generic`**.

5.4.2.1. Liste des erreurs locales

- ASYNCHRONOUS_VIOLATION

  Cette erreur se produit en cas de demande de verrouillage, de déverrouillage ou de vérification d'existence d'instances dépendantes en mode asynchrone.
- CARDINALITY_VIOLATION

  Cette erreur se produit en cas de non respect des cardinalités lors de l'activation d'une méthode de mise à jour.
- CURRENT_INSTANCE_MISSING

  Cette erreur se produit quand une méthode est appliquée à la propriété **`detail`** alors que cette propriété ne contient pas d'instance.
- CURRENT_USER_INSTANCE_MISSING

  Cette erreur se produit quand une méthode utilisateur est appliquée à la propriété **`userDetail`** alors que cette propriété ne contient pas d'instance.
- FOLDER_USER_CONTEXT_LENGTH_ERROR

  Cette erreur se produit quand le contenu d'une rubrique du buffer utilisateur a une taille supérieure à la taille autorisée pour la rubrique.
- INSTANCE_ALREADY_LOCKED

  Cette erreur se produit en cas de demande de verrouillage d'une instance déjà verrouillée sur le serveur.
- INSTANCE_NOT_LOCKED

Cette erreur se produit en cas de demande de déverrouillage sur une instance non verrouillée sur le serveur.

- INVALID_CHANGE

  Cette erreur se produit lorsque l'instance à modifier n'existe pas dans le cache local.

- INVALID_CREATION

  Cette erreur se produit en cas de création d'une instance qui existe déjà dans le cache local.

- INVALID_DELETION

  Cette erreur se produit lorsque l'instance à supprimer n'existe pas dans le cache local.

- INVALID_INITIALIZATION

  Cette erreur se produit dans le cas d'une tentative d'initialisation d'une instance déjà connue du cache local quel que soit son statut (READ, CREATED, MODIFIED ; DELETED).

- INVALID_INSTANCE

  Cette erreur se produit lorsqu'une clé primaire de l'instance courante n'est pas valide.

- LENGTH_ERROR

  Cette erreur se produit quand le contenu d'une rubrique de l'instance courante a une taille supérieure à la taille autorisée pour la rubrique.

- PARENT_INSTANCE_MISSING

  Cette erreur se produit en cas de sélection d'une instance de noeud dépendant alors que l'instance supérieure est inexistante.

- REFERENCE_USER_CONTEXT_LENGTH_ERROR

  Cette erreur se produit quand la taille du contenu d'une Rubrique du buffer utilisateur associé à un nœud référence est supérieure à la taille autorisée pour la Rubrique.

- REFERING_INSTANCE_MISSING

  Cette erreur se produit quand la méthode `transferReference` ne trouve pas d'instance dans le `detail` du noeud référençant.

- SERVER_UPDATE_REQUIRED

  Cette erreur se produit lorsqu'une méthode est appliquée à une instance alors que l'instance supérieure créée localement n'existe pas encore dans la base. Une mise à jour serveur préalable de l'instance supérieure est requise.

- SUBSCHEMA_ERROR

  Cette erreur se produit lorsqu'une rubrique de l'instance courante est modifiée alors qu'elle n'a pas été renseignée pour cette instance à l'issue d'un accès serveur paramétré avec un sous-schéma.

- UNKNOWN_ASYNCHRONOUS_REQUEST_ID

  Cette erreur se produit lorsqu'une demande de récupération de réponse a été émise et que l'identifiant de réponse est inconnu ou a dépassé le délai d'expiration.

- UNKNOWN_INSTANCE

Cette erreur se produit en cas de sélection d'une instance inconnue du cache local.

- VALUE_ERROR

  Cette erreur se produit quand le contenu d'une rubrique de l'instance courante n'est pas valide.

- VALUE_REQUIRED

  Cette erreur se produit quand une rubrique de l'instance courante est considérée comme non présente alors qu'elle est obligatoire.

☞ Ces intitulés d'erreurs correspondent à des constantes de la classe **`com.ibm.vap.generic.LocalException`** et représentent des types d'erreurs. Préfixé par **`LOCAL_`**, chaque constante détermine une clé d'erreur. Cette clé d'erreur permet d'identifier le libellé d'erreur associé dans le fichier local des libellés d'erreurs **`vaperror.properties`**.

5.4.3. Erreurs serveur

Les erreurs serveur provoquent l'exception **`com.ibm.vap.generic.ServerException`**.

Cette exception est levée à la suite de la réception d'un message d'erreur logique détectée par le composant Serveur. Elle porte la clé et le libellé associés.

5.4.3.1. Structure de la clé d'erreur

Vous devez connaître la clé d'erreur si vous souhaitez afficher des libellés d'erreur personnalisés dans le Client.

☞ Pour des informations sur la manière dont sont déterminées les clés d'accès aux libellés locaux des erreurs serveur, reportez-vous au *Manuel de Référence Clients Graphiques : Interface Publique des composants générés*.

Col 1-3	Col 4-9	Col 10-13	Col 14-19	Col 20	Col 21	Col 22-25	
bib	ser	seg			E	DUPL	Création à tort
bib	ser	seg			E	NFND	Modif/annul à tort
bib	ser				E	code erreur	Erreur utilisateur
bib	ser	vue	rub	2	E		Rubrique obligatoire
bib	ser	vue	rub	5	E		Erreur de valeur
bib	ser		LOCKED		E		Déjà verrouillée
bib	ser		NTLOCK		E		Déverrouillage à tort

Légende bib = code bibliothèque vue = code Vue Logique ser = code serveur

rub = code Rubrique seg = code segment d'accès physique

E = Exception (Erreur serveur)

5.4.4. Erreurs système

Les erreurs système (physiques) provoquent l'erreur **`com.ibm.vap.generic.SystemError`**. Ce type d'erreur représente une erreur interne et irrécupérable.

5.4.4.1. Structure de la clé d'erreur

Vous devez connaître la clé d'erreur si vous souhaitez afficher des libellés d'erreur personnalisés dans le Client.

Pour des informations sur la manière dont sont déterminées les clés d'accès aux libellés locales des erreurs système, reportez-vous au *Manuel de Référence Clients Graphiques : Interface Publique des composants générés*.

5.4.4.1.1. Erreurs système reçues du Composant Applicatif

Dans le tableau suivant, vous trouvez la liste des erreurs système renvoyées par le Composant Applicatif.

Col 1-3	Col 4-9	Col 10-13	Col 14-19	Col 20	Col 21	Col 22-25	
bib	ser		LKABSC		S		Timestamp non défini pour verrouillage
bib	ser	vue	ACCESS		S		Erreur d'accès données
bib	ser	STRU			S		Erreur structure vue
bib	ser	VERS			S		Erreur version
bib	ser	VIEW			S		Vue inconnue
bib	ser	SERV			S		Service inconnu
bib	ser	LTH			S		Longueur vue erronée
			MISPCV		S		Déphasage composants

Légende bib = code bibliothèque vue = code Vue Logique ser = code serveur

rub = code Rubrique seg = code segment d'accès physique

S = Erreur système

L'erreur de type **`Service inconnu`** apparaît lorsque le service demandé par la Proxy n'est pas reconnu par le Composant Applicatif.

L'erreur de type **`Longueur de vue erronée`** apparaît lorsqu'il y a un changement de format dans une Vue Logique associée à la Proxy et que celle-ci n'a pas été regénérée.

Pour résoudre ce problème, vous devez regénérer la Proxy.

L'erreur de type **`Déphasage composants`** survient lorsqu'il y a un déphasage entre les composants Client et Serveur.

Pour résoudre ce problème, vous devez regénérer la Proxy.

5.4.4.1.2. Erreurs système reçues du Moniteur de Communication

Ces erreurs sont, pour la plupart, des erreurs internes que vous résoudrez en contactant le support VisualAge Pacbase.

Col 1-3	Col 4-9	Col 10-13	Col 21	
bib	mon	LSRV	S	Erreur de longueur du message reçu
bib	mon	PNUM	S	Erreur de structure du paramètre "numéro de service"
bib	mon	PCOD	S	Erreur de structure du paramètre "code de service"
bib	mon	PNOD	S	Code du Dossier inconnu du Moniteur de Communication
bib	mon	PCVS	S	Erreur de structure d'une demande de service en provenance du client
bib	mon	PCVF	S	Erreur de structure du message en provenance du client
bib	mon	TAND	S	Erreur Tandem/Pathway
bib	mon	WF00	S	Erreur d'accès au fichier de travail ou à la base de données (open/close)

Légende bib = code bibliothèque mon = code Moniteur de Communication S = erreur système

5.4.4.1.3. Erreurs système reçues du Gestionnaire de Services

Ces erreurs sont, pour la plupart, des erreurs internes que vous résoudrez en contactant le support VisualAge Pacbase.

Col 1-3	Col 4-9	Col 10-13	Col 21	
bib	ges	SRV1	S	Service non trouvé dans le fichier de travail
bib	ges	LNG1	S	Erreur de conversion de la longueur du service sur une requête multi-messages
bib	ges	LNG2	S	Erreur de conversion de la longueur du service sur une requête mono-message
bib	ges	NOS1	S	Erreur de structure du paramètre "numéro de service"
bib	ges	NOS2	S	Erreur de conversion du paramètre "numéro de service"
bib	ges	SRV1	S	Erreur de structure du paramètre "code service"
bib	ges	SRV2	S	Code service inconnu dans le Dossier
bib	ges	DOS1	S	Paramètre "nom du Dossier" absent
bib	ges	DOS2	S	Erreur de longueur du paramètre "nom du Dossier"
bib	ges	VER2	S	Erreur de longueur du pramètre "numéro de version"
bib	ges	NOD1	S	Paramètre "nom du noeud" absent
bib	ges	NOD2	S	Erreur de longueur du paramètre "nom du noeud"
bib	ges	NOD3	S	Nom du noeud inconnu dans le Dossier
bib	ges	TYNO	S	Service non autorisé sur le noeud
bib	ges	SCH1	S	Erreur de structure du paramètre "code sous-schéma"
bib	ges	SCH2	S	Code sous-schéma inconnu dans le Dossier
bib	ges	NOCP	S	Erreur de structure du paramètre "nombre d'occurrences"
bib	ges	NOC1	S	Erreur de longueur sur le paramètre "nombre d'occurrences"
bib	ges	NOC2	S	Erreur de conversion du paramètre "nombre d'occurrences"
bib	ges	EXT1	S	Erreur de structure du paramètre "méthode d'extraction"
bib	ges	EXT2	S	Méthode d'extraction inconnue dans le Dossier
bib	ges	USR1	S	Paramètre "service utilisateur" absent
bib	ges	USR2	S	Erreur de longueur sur le paramètre"service utilisateur"
bib	ges	USR3	S	Service Utilisateur inconnu dans le Dossier
bib	ges	CHK1	S	Paramètre "Check Option" absent
bib	ges	CHK2	S	Erreur de longueur sur le paramètre "Check Option"
bib	ges	RFH1	S	Paramètre "Refresh Option" absent
bib	ges	RFH2	S	Erreur de longueur sur le paramètre"Refresh Option"
bib	ges	LCK1	S	Paramètre "Lock Timestamp" absent
bib	ges	LCK2	S	Erreur de longueur sur le paramètre "Lock Timestamp"
bib	ges	PCV1	S	Erreur de structure du paramètre "Selection Criteria"
bib	ges	PC01	S	Erreur de conversion du paramètre "Selection Criteria"
bib	ges	PILO	S	Erreur d'accès à l'enregistrement pilote du fichier de travail
bib	ges	BUF1	S	Erreur de structure du buffer utilisateur
bib	ges	PC02	S	Erreur de conversion d'un champ du buffer utilisateur
bib	ges	FRWR	S	Erreur d'accès en écriture au fichier de travail
bib	ges	FRW2	S	Erreur d'écriture du dernier enregistrement du fichier de travail
bib	ges	FRRE	S	Erreur d'accès en lecture au fichier de travail avant mise à jour
bib	ges	FRRD	S	Erreur d'accès en lecture au fichier de travail

Col 1-3	Col 4-9	Col 10-13	Col 21	
bib	ges	FRRW	S	Erreur d'accès en mise à jour au fichier de travail
bib	ges	CP01	S	Erreur de structure d'un champ de "Selection Criteria" en provenance du Composant Applicatif
bib	ges	CP02	S	Erreur de structure d'un champ d'une instance de Vue Logique en provenance du Composant Applicatif
bib	ges	ERKY	S	Erreur de structure de la clé du "Selt Message" en provenance du Composant Applicatif
bib	ges	ERLA	S	Erreur de structure du label du "Selt Message" en provenance du Composant Applicatif
bib	ges	PCV!	S	Erreur de structure d'un champ du buffer utilisateur en provenance du Composant Applicatif
bib	ges	PCV3	S	Erreur de structure d'un champ d'une instance de Vue Logique en provenance du client
bib	ges	PC03	S	Erreur de conversion d'un champ d'une instance de Vue Logique en provenance du client
bib	ges	PCV4	S	Erreur de structure d'un champ de "Selection Criteria" en provenance du client
bib	ges	PC05	S	Erreur de conversion d'un champ de "Selection Criteria" en provenance du client
bib	ges	NUVE	S	Erreur du paramètre "numéro de version" dans le Composant Applicatif élémentaire
bib	ges	STRU	S	Erreur du paramètre "structure" dans le Composant Applicatif élémentaire
bib	ges	VIEW	S	Erreur du paramètre "code de la Vue Logique" dans le Composant Applicatif élémentaire
bib	ges	SERV	S	Erreur du paramètre "code opération" dans le Composant Applicatif élémentaire
bib	ges	LTH	S	Erreur du paramètre "longueur" dans le Composant Applicatif élémentaire
bib	ges	PCV5	S	Erreur de structure du code action d'une instance de Vue Logique à mettre à jour
bib	ges	PCV6	S	Erreur de structure d'un champ d'une instance de Vue Logique à mettre à jour en provenance du client
bib	ges	PCV7	S	Erreur de structure du vecteur de présence d'un champ d'une instance de Vue Logique à mettre à jour
bib	ges	PC06	S	Erreur de conversion d'un champ d'une instance de Vue Logique à mettre à jour en provenance d'un Composant Applicatif élémentaire
bib	ges	ERK1	S	Erreur de structure de la clé de message d'erreur utilisateur
bib	ges	ERL1	S	Erreur de structure du label de message d'erreur utilisateur
bib	ges	OCNB	S	Erreur de structure, dans le message d'erreur utilisateur, du code champ sur lequel porte l'erreur
bib	ges	DANA	S	Erreur de structure du code de champ erroné dans le message d'erreur utilisateur
bib	ges	PCVS	S	Erreur de structure d'une demande de service en provenance du client
bib	ges	PCVF	S	Erreur de structure du message en provenance du client
bib	ges	WF00	S	Erreur d'accès au fichier de travail
bib	ges	TAND	S	Erreur Tandem/Pathway

Légende bib = code bibliothèque ges = code Gestionnaire de services S = erreur système

5.4.5. Erreurs de communication

Les erreurs dans la chaîne de communication avec le Serveur provoquent l'erreur `com.ibm.vap.generic.CommunicationError`. Comme toute erreur Java, une erreur de communication VisualAge Pacbase ne renvoie pas de clé. Pour identifier la cause de l'erreur, il faut récupérer le message associé à l'erreur (méthode `getMessage()` de la classe).

5.4.5.1. Liste des erreurs

Trois messages d'erreur de communication sont susceptibles de s'afficher :

- `Erreur open serveur`
- `Erreur appel serveur`
- `Erreur close serveur`

Si un message d'erreur de communication apparaît, vous devez d'abord en informer le responsable de la communication car la cause peut être un encombrement de ligne, une ligne défectueuse, un serveur indisponible...

5.4.6. Exemple de gestion des erreurs

5.4.6.1. Introduction

Le code Java de cet exemple sera fourni à la livraison de la version Java de Pacbench C/S. Il illustre une méthode qui permet de gérer tous les types d'erreurs susceptibles d'être levées par les objets Proxy. Pour cela, trois classes ont été définies :

- `StandardErrorMessageFactory` : classe permettant de créer un vecteur de `StandardErrorMessage` à partir de l'exception levée
- `StandardErrorMessage` : classe unifiant les caractéristiques des différents types d'erreurs
- `ErrorManagerExample` : classe graphique utilisée pour afficher les erreurs

5.4.6.2. Présentation des classes non visuelles utilisées par l'exemple

- Classe `StandardErrorMessageFactory`

Cette classe offre la méthode `public static java.util.Vector getStandardErrorMessages (Throwable th)` qui permet de créer un vecteur de `StandardErrorMessage` à partir d'une exception donnée quel que soit son type.

- Classe `StandardErrorMessage`

Chaque objet de type `StandardErrorMessage` a des propriétés qui sont initialisées selon le type d'erreur (locale, serveur, système, communication, etc.) :

- la **Proxy hiérarchique associée à l'erreur** si l'erreur concerne une instance de Vue Logique particulière.
- la **clé de l'erreur** (pour les erreurs locale, serveur ou système) :

C'est une chaîne de caractères retournée par le serveur, dans le cas d'une erreur serveur ou système.

Dans le cas d'une erreur locale, c'est une chaîne de caractères correspondant au nom de la constante associée au type de l'erreur et définie dans la classe **`LocalException`**, préfixé par **`LOCAL_`**.

Pour connaître tous les types d'erreurs locales, reportez-vous à la section **5.4.2.1**.

Pour connaître les clés d'erreur correspondant aux erreurs serveur et système, reportez-vous respectivement à la section **5.4.3**, *Erreurs serveur* et **5.4.4**, *Erreurs système*.

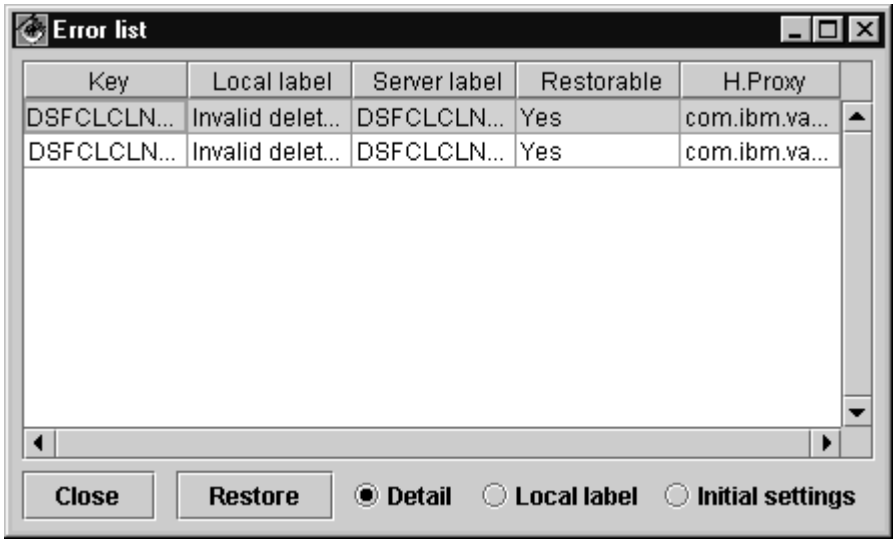
- le **libellé local de l'erreur** :
  - dans le cas d'une erreur locale, serveur ou système, le libellé résulte de la traduction par le Client de la clé de l'erreur. Dans le fichier **`vaperror.properties`**, une table de correspondance permet d'identifier ce libellé à partir d'une clé donnée.

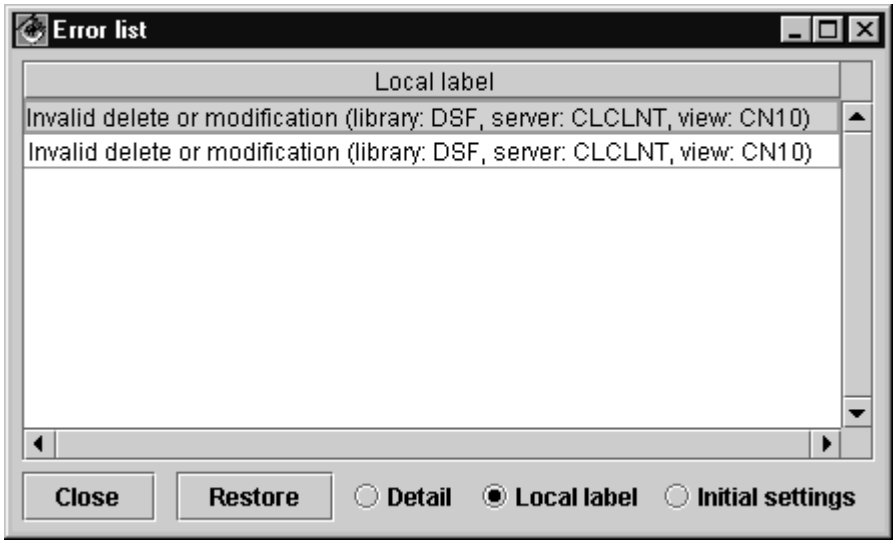
    Pour des informations sur le fichier **`vaperror.properties`**, reportez-vous au *Manuel de Référence Clients Graphiques : Interface Publique des composants générés*.
  - Pour les autres types d'erreurs, le libellé correspond directement au message contenu dans l'exception Java.
- le **libellé serveur de l'erreur**, pour les erreurs serveur et système, si un serveur de libellés d'erreurs a été codé dans les Services Applicatifs.
- la propriété booléenne **`Restorable`**. Cette propriété est vraie :
  - si une Proxy est associée à l'erreur.
  - s'il est possible de réafficher l'instance à l'origine de l'erreur dans le détail de la fenêtre qui appelle la Proxy.

Pour des informations complémentaires sur la gestion des erreurs, reportez-vous au *Manuel de Référence Clients Graphiques : Interface Publique des composants générés*.

5.4.6.3. Présentation de la classe visuelle ErrorManagerExample

5.4.6.3.1. Interface graphique





5.4.6.3.2. Fonctionnalités de la classe

Cette classe est utilisée pour présenter une liste de messages d'erreur correspondant à des instances de la classe **`StandardErrorMessage`**.

Tous les contrôles graphiques utilisés dans cette fenêtre sont proportionnels à sa taille.

Les fonctionnalités de cette fenêtre sont les suivantes :

- Les instances de **`StandardErrorMessage`** calculées par la classe **`StandardErrorMessageFactory`** sont passées à cette fenêtre dans un Vector par l'intermédiaire de la méthode **`addStandardErrorMessages(Vector)`**. Tant que la fenêtre n'est pas fermée, les messages d'erreur passés sont cumulés aux messages d'erreur déjà affichés. Les messages d'erreur affichés peuvent être supprimés en utilisant la méthode **`resetCurrentStandardErrorMessages()`**.

- Les propriétés des instances de **`StandardErrorMessage`** que l'on souhaite visualiser sont définies dans l'attribut **`visibleColumns`** des propriétés de la fenêtre **`ErrorManagerExample`**. Cet attribut correspond à un tableau de **`int`** qui peuvent prendre les valeurs suivantes :
  - **`1`** pour visualiser la clé du message d'erreur
  - **`2`** pour visualiser le libellé local du message d'erreur
  - **`3`** pour visualiser le libellé serveur du message d'erreur
  - **`4`** pour visualiser le statut de restauration de l'erreur
  - **`5`** pour visualiser le nom de la classe associé à la Proxy hiérarchique

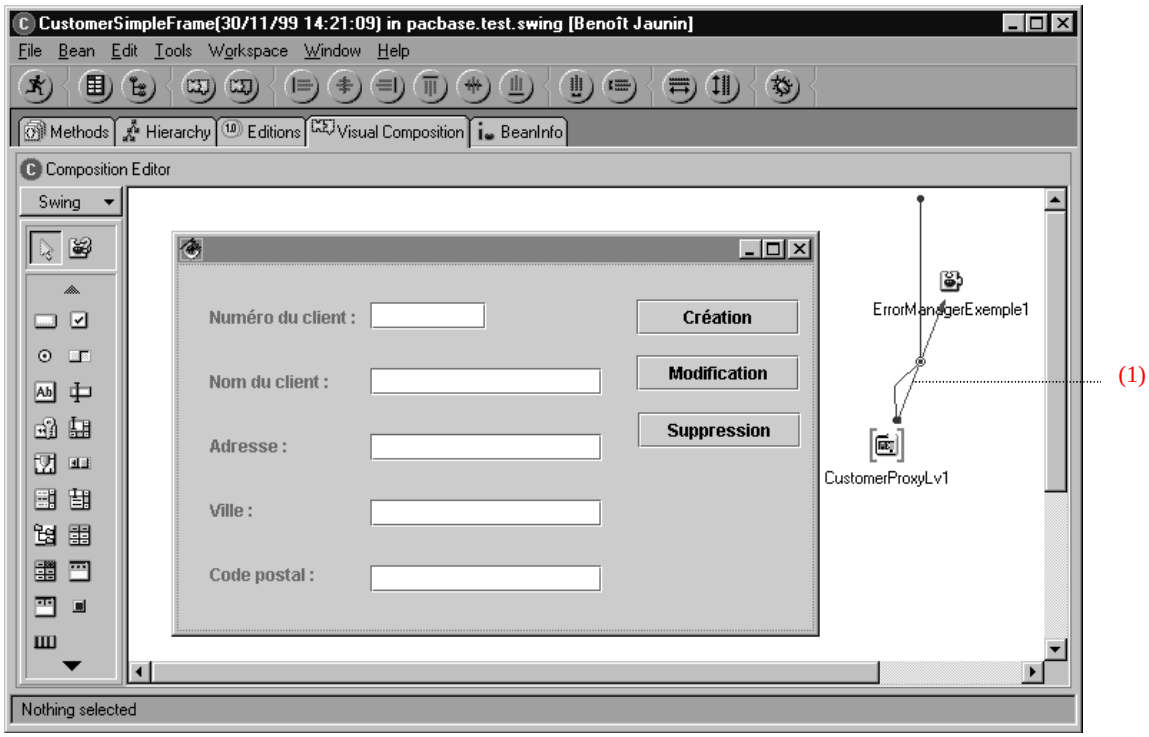
- Trois boutons radios ont été programmés pour permettre de sélectionner dynamiquement des présentations différentes de la liste des instances de **`StandardErrorMessage`** :
  - **`Detail`** pour visualiser toutes les propriétés de chaque instance de **`StandardErrorMessage`**.
  - **`Local label`** pour ne visualiser que le libellé local de chaque instance de **`StandardErrorMessage`**.

- **Initial settings** pour visualiser les propriétés définies par l'attribut **visibleColumns**.

- Le bouton **Restore** est activé lorsqu'une instance de **StandardErrorMessage** est sélectionnée et qu'il est possible de restaurer le contexte de l'erreur. Lorsque l'utilisateur clique sur ce bouton, le contexte de l'erreur est restauré et la fenêtre qui gère l'instance de la Vue Logique qui a provoqué l'erreur est affichée.

  Pour mettre en œuvre cette fonctionnalité, il est nécessaire de faire connaître à la fenêtre **ErrorManagerExample,** chaque fenêtre gérant l'attribut **detail** du nœud d'une Proxy en utilisant la méthode **addProxyManagingbyWindow**. Cette méthode prend en paramètres le nœud de la Proxy (**HierarchicalProxyLv**) et la fenêtre de gestion des erreurs (**Jframe**).

Exemple



Cet exemple illustre le principe qui permet d'enregistrer l'association d'une Proxy hiérarchique et la fenêtre qui l'utilise et de faire connaître cette association à la fenêtre d'erreur.

Le lien (1) connecte l'événement **this** de la Proxy à la méthode **addProxyManagingbyWindow** de la fenêtre d'erreurs (**ErrorManagerExample**). Dans ce contexte, cette méthode sera exécutée dès l'instanciation de la Proxy.

Les deux autres liens permettent de passer l'instance de la Proxy hiérarchique (**this** de la Proxy sur paramètre **hp** de la connexion) et l'instance de la fenêtre (**this** de **CustomerSimpleFrame** sur paramètre **wi** de la connexion).

5.4.6.4. Code pour l'affichage de la fenêtre d'erreur

La méthode proposée est adaptée aux applications conçues dans VisualAge Java à l'aide de l'outil de composition visuelle.

Cette méthode suppose que la fenêtre de gestion des erreurs (classe **`ErrorManagerExample`**) a été insérée en tant que bean de type *class* ou *variable* dans l'éditeur de composition visuelle qui contient la fenêtre qui l'appelle.

Les exceptions associées à cette fenêtre sont traitées en insérant dans la méthode **`handleException( Throwable)`** de celle-ci le code suivant :

```
private void handleException(Throwable exception) {
java.util.Vector standardErrorMessages =
pacbase.test.swing.ErrorManager.StandardErrorMessageFactory.getStandardErrorMessages(exception);
  if (standardErrorMessages != null) {
        if (standardErrorMessages.size() >= 0) {
              getErrorManagerExemple1().addStandardErrorMessages(standardErrorMessages);
              getErrorManagerExemple1().showStandardErrorMessages();
        }
  }
}
```

5.5. Gestion de la communication

5.5.1. Composants du middleware

Le middleware de Pacbench Client/Serveur correspond à une application C et C++ dont les fonctions sont réparties dans trois types de DLLs :

- le premier type regroupe les fonctions d'interfaçage avec les langages C et C++ et les fonctions d'interprétation des services du middleware et d'aiguillage vers des librairies de fonctions spécialisées pour un protocole de communication spécifique.
- le deuxième type regroupe les fonctions d'exécution des services du middleware de Pacbench Client/Serveur pour un protocole de communication spécifique.
- le troisième type regroupe des fonctions de gestion de ressources telles que la traduction en clair des exceptions rencontrées.

La version OS/2 du middleware de Pacbench Client/Serveur utilise également un run-time matérialisé par une DLL de code **CSOOM30**.

Nom de la DLL	Fonction	Plate-forme	Protocole
ixomwarp.dll	Type-1	Windows 95/NT OS2	Tous
ixomware.dll	Type-1	Windows 95/NT OS2, UNIX	Tous
ixocics.dll	Type-2	Windows 95/NT OS2	CICS-ECI
ixocpic.dll	Type-2	Windows 95/NT OS2	CPI-C
ixotux.dll	Type-2	Windows 95/NT UNIX	TUXEDO
ixosock.dll	Type-2	Windows 95/NT OS2, UNIX	TCP-IP Socket VA Pacbase
ixoloc.dll	Type-2	Windows 95/NT OS2	Appel de DLL COBOL
ixomqs.dll	Type-2	Windows 95/NT OS2, UNIX	MQSERIES
ixotmvs.dll	Type 2	Windows 95/NT, OS2, UNIX	TCP-IP Socket CICS
ixotcis.dll	Type 2	Windows 95/NT, OS2, UNIX	TCP-IP Socket TCIS
ixomsgen.dll	Type-3	Windows 95/NT OS2	Tous
csoom30.dll	runtime C++	OS2	Tous

Windows(*) = toutes plate-formes Windows

5.5.2. Traitement d'une requête

Dans VisualAge, les services du middleware sont exécutés à partir d'un ensemble de classes de communication spécifiques livrées à l'installation du produit.

La communication avec le Serveur s'effectue via l'interface **`ServerAdapter`**. Deux implémentations de cette interface existent :

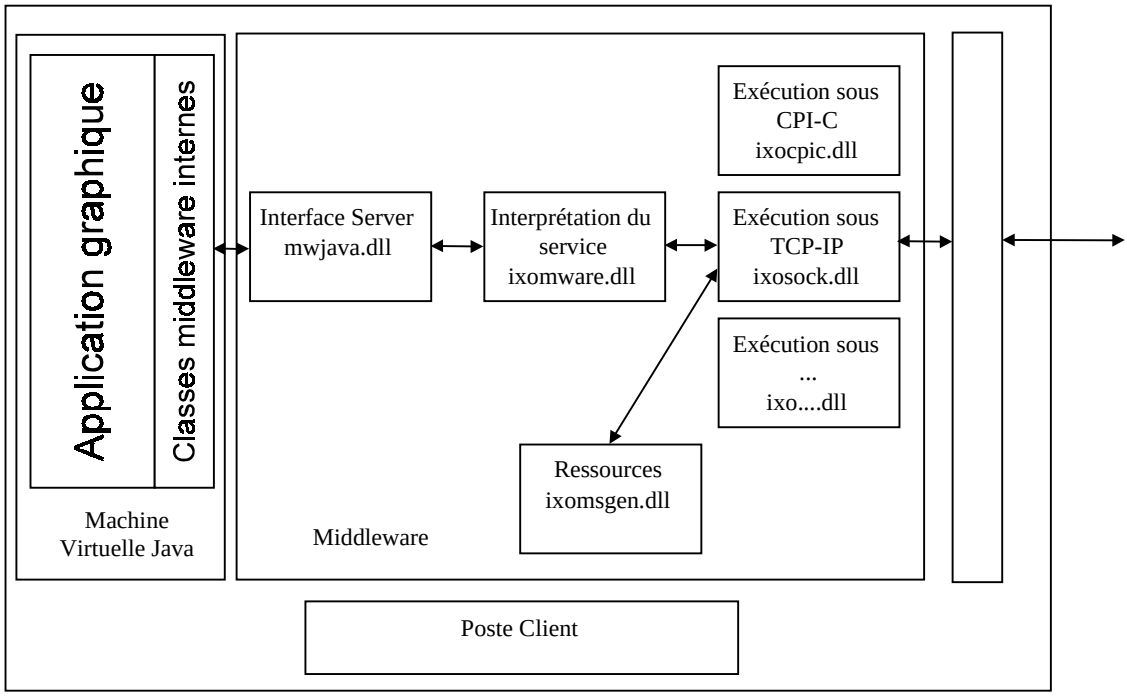
- une implémentation faisant un accès direct aux DLL C++ du middleware. Cette implémentation est dédiée aux applications standard.

Les DLL doivent être installées dans le path.

- une implémentation faisant appel à une Gateway ou un Relais via TCP/IP. Cette implémentation est dédiée aux applets.

5.5.2.1. Accès direct au middleware ou mode local

5.5.2.1.1. Schéma de traitement d'une requête



5.5.2.1.2. Instanciation d'un Dossier en mode local

Afin d'instancier un Dossier en mode local, vous pouvez utiliser le constructeur par défaut : **`CustomerProxyLv()`**

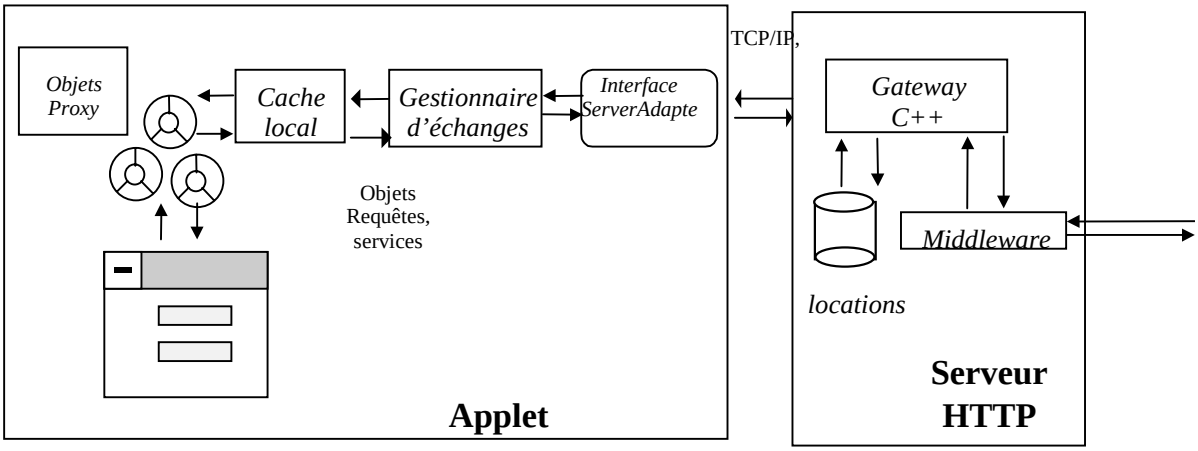
Par exemple :

```
myProxy = new CustomerProxyLv() ;
myproxy.setLocationsFile("c:\vap\gen\java\VAPLOCAT.INI") ;
```

La seconde instruction spécifie le fichier de locations à utiliser. Par défaut, on recherche un fichier **`VAPLOCAT.INI`** se trouvant dans le répertoire courant.

5.5.2.2. Accès via une Gateway

5.5.2.2.1. Schéma de traitement d'une requête



5.5.2.2.2. Utilisation de la Gateway et du Relais

La gateway et le relais sont utilisés par les objets Proxy pour accéder au serveur de dossier depuis une applet VisualAge Java.

- **Utilisation de la Gateway**

La gateway est le programme chargé de faire les accès au middleware.

Elle se lance avec la commande **`gateway.exe`**. Sa syntaxe est la suivante :

```
gateway [-h] –s|i|d TCP_PORT_NUM [-l LOCATION_FILE] [-t[TRACE_LEVEL]]
[-c CHAR_CONV_FILE] [-pcv]
```

où :

- `-h[elp]` : option qui permet d'afficher la syntaxe
- `-s` : lance la gateway
- `-i` : installe la gateway comme un service Windows NT

  Par défaut, la gateway est installée en mode de démarrage automatique.
- `-d` : désinstalle la gateway des services Windows NT
- `TCP_PORT_NUM` : positionne le port de communication utilisé par les objets Proxy.
- `-l LOCATION_FILE` : spécifie le fichier des *locations*. Defaut : « .\VAPLOCAT.INI ».
- `-t[TRACE_LEVEL]` : positionne le niveau de trace.
  - 0 : muet (défaut)
  - 1 : erreurs
  - 2 : standard
  - >2 : debug
- `-c CHAR_CONV_FILE` : spécifie le fichier de conversion du code page
- `-pcv` : Optimisation de la longueur des messages pour le PCV. Par la gateway, cette option est obligatoire.

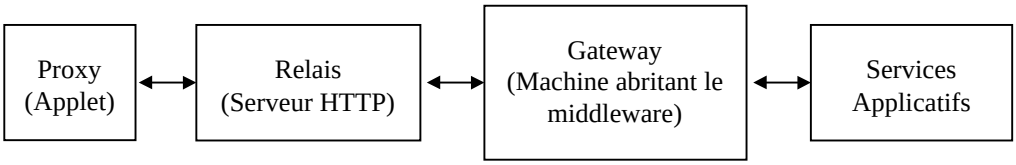
Par exemple :

```
gateway –s 6001 -l c:\vap\generated\java\VAPLOCAT.INI -t1 –pcv
```

- **Utilisation du Relais**

Le relais est utilisé pour relayer les requêtes arrivant à un serveur HTTP vers une autre machine qui abrite le middleware et la gateway.

En voici une utilisation standard :



Il peut également être utilisé dans le cadre d'Internet. Il se lance avec la commande **`relay`**.

Sa syntaxe est la suivante :

```
relay [-h] [CLIENT_PORT] [-t[TRACE_LEVEL]] -server HOST[:PORT]
```

où :

- `-h[elp]` : est l'option qui permet d'afficher la syntaxe
- `[CLIENT_PORT]` : positionne le port de communication. Défaut : 5647
- `-t[TRACE_LEVEL]` : niveau de trace (ce sont les mêmes valeurs que celles de la gateway)
- `-server HOST[:PORT]` : adresse de la gateway/relais qu'il faut relayer.

Par exemple :

```
relay 5647 -t1 -server pc12:6001
```

5.5.2.2.3. Instanciation d'un Dossier par la gateway

Deux constructeurs peuvent dans ce cas être utilisés :

- **`CustomerProxyLv(String host)`**

  Par exemple : `new CustomerProxyLv("9.134.5.146")`
- **`CustomerProxyLv(String host, int port)`**

  Par exemple : `new CustomerProxyLv("9.134.5.146", 6001)`

☞ Si **`host`** vaut null, on considère que le middleware est local.

5.5.2.3. Changement dynamique des paramètres d'accès au middleware

Plusieurs méthodes de la Proxy Racine permettent de changer ces paramètres dynamiquement :

- **`void setHost(String host)`**

☞ Si **`host`** vaut null, on considère que le middleware est local.

- **`void setPort(int port)`**
- **`void setLocationsFile(String filename)`** (mode local)
- **`void setTraceFile(String filename)`**
- **`void setTraceLevel(int traceLevel)`** (mode local)

5.5.3. Définition du contexte d'utilisation

La gestion de la communication nécessite la définition du contexte d'utilisation du middleware. Ce contexte est défini dans un fichier spécifique dans lequel sont indiqués, pour chaque Dossier généré, une ou plusieurs localisations.

Une localisation, ou *location* permet de spécifier les éléments qui autorisent les échanges de données entre les applications clientes et les gestionnaires de services et ce, que ce soit dans une étape de développement de l’application cliente ou lors de son exploitation. Par exemple, une application cliente peut s’adresser à un serveur sans trace pour une version d’exploitation.

Pour un Dossier, une localisation est composée :

- de l’identifiant de la localisation. Cet identifiant est sensible à la distinction majuscules/minuscules.
- du nom externe du moniteur de communication.
- de la longueur physique maximum d’un message d’échange.
- d’un nombre variable de paramètres qui permettent de faire fonctionner le middleware (adresse IP, trace...).

Les localisations sont spécifiées dans le fichier **`VAPLOCAT.INI`** (ce nom de fichier en majuscules est réservé).

5.5.3.1. Structure du fichier VAPLOCAT.INI

Le fichier des localisations est structuré en sections et sous-sections.

Une section correspond à un Dossier et est identifiée par le code de celui-ci, placé entre [...].

Une sous-section correspond à une localisation et est identifiée par le nom de l'environnement spécifié pour l'option **`LOCATION`** dans la fenêtre **`Commentaires`** du Dossier. L'identifiant d'une sous-section est placé entre <...>.

Dans une sous-section, les différents paramètres sont représentés par des mots-clés.

Les mots-clés utilisés sont les suivants :

Mot-clé	Signification
COMMENT	Commentaire
LENGTH	Longueur physique du message (générée)
MONITOR	Nom externe du moniteur de communication (généré)
MWADDRESS	Adresse du serveur (pour SOCKET, TCPMVS, TCIS, TCPIMSI et TCPIMSE) Obligatoire pour ces protocoles Les formats suivants sont possibles: - Codification décimale : * adresse IP ou alias de l'adresse IP * blanc * numéro de port en décimal - Codification hexadécimale: * [1, 2]: '0x'' * [3, 6]: domaine AF_INET * [7,10]: numéro de port en hexadécimal * [11, 18]: adresse IP
MWARE	Protocole de communication **Valeurs** : **Protocoles** TUXEDO TUXEDO XA ou non XA CICS CICS ECI extend ou nonextend LOCAL Serveur sur la station Client CPIC CICS CPI-C/APPC ou IMS CPI-C/APPC SOCKET TCP-IP SOCKET pour UNIX ou Windows/NT TCIS TCIS MQSERIES MQSERIES TCPMVS TCP-IP SOCKET pour MVS/CICS CPICXCP2 CPI-C/XCP2 pour TDS ou TP8
MWBUFFERTYPE	Type de buffer (pour TUXEDO) Valeurs : - CARRAY (défaut) - FML
MWCODEPAGE	Code page du serveur - Facultatif
MWFMLNAME	Nom de fichier FML (pour TUXEDO). Attention : vous devez aussi indiquer le chemin dans la variable d'environnement FLDTBLDIR et le nom du fichier dans la variable d'environnement FIELDTBLS.
MWMAXREPLY	Nombre maximum de requêtes asynchrones en attente de réponse - Facultatif - Numérique
MWQUEUEMANAGER	Nom du manager de queues pour protocole asynchrone (MQSERIES) - Obligatoire pour ce protocole - Alphanumérique - 48 caractères max.
MWREPLYQUEUE	Nom de la queue de message en réception pour protocole asynchrone (MQSERIES) - Obligatoire pour ce protocole - Alphanumérique - 48 caractères max.
MWREQUESTEXPIRY	Durée de vie d'une requête asynchrone (en secondes) – Facultatif (illimité par défaut) - Numérique

Mot-clé	Signification
MWREQUESTQUEUE	Nom de la queue de message en émission pour protocole asynchrone (MQSERIES) - Obligatoire pour ce protocole - Alphanumérique - 48 caractères max.
MWTIMEOUT	Time out géré par le middleware (en secondes) - Facultatif - Numérique - Valeur: [0, 32767]
MWTRANSID	Code de transaction CICS sur 4 caractères : - obligatoire pour le protocole TCP-IP socket MVS - facultatif pour le protocole CICS/ECI • Si le paramètre MWTRANSID est absent ou non valorisé, la transaction CPMI est automatiquement exécutée sur CICS. Dans ce cas, le paramètre MONITOR est obligatoire et doit correspondre à un programme exécutable sous CICS. • Si le paramètre est valorisé, le code de transaction sera exécuté sous CICS pour chaque requête utilisant cette location. Dans ce cas, la valeur du paramètre MONITOR n'est pas exploitée.

Chaque ligne de fichier ne doit contenir qu'un paramètre.

Exemple de fichier de localisation :

```
[FOCLNT]
<Location1>
COMMENT=moniteur d'exploitation
MONITOR=CLCOMM
LENGTH=8192
MWCODEPAGE=850
MWARE=SOCKET
MWADDRESS=0x00021770C0060A5D
<Location2>
COMMENT=moniteur avec trace
MONITOR=CLCOM2
LENGTH=8192
MWARE=SOCKET
MWADDRESS=0x00021770C0060A5D
<Location3>
COMMENT=moniteur local
MONITOR=CLCOMM
LENGTH=8192
MWARE=LOCAL
```

5.5.3.2. Mise en oeuvre

Le fichier **VAPLOCAT.INI** est obligatoire en phase de développement et en phase d'exploitation.

En cas d'absence de ce fichier ou si la syntaxe en est incorrecte, le Gestionnaire d'échanges ne peut être instancié. Pour connaître le paramètre erroné, consultez le fichier trace.

En phase de développement, le générateur de Proxy Vues de Dossier assure la création ou la mise à jour du fichier **VAPLOCAT.INI**.

- S'il n'existe pas, le générateur le crée dans le répertoire courant de VisualAge.

  Pour chaque localisation, les valeurs des paramètres **MONITOR** et **LENGTH** (générés) sont automatiquement positionnées.

  Le développeur a la charge de positionner les autres paramètres.
- S'il existe déjà (dans le répertoire de VisualAge), il est automatiquement chargé. Pour chaque Dossier, le générateur procède alors à une comparaison entre les localisations du fichier d'extraction et les localisations existantes. Il assure la mise à jour du fichier existant :
  - en ajoutant automatiquement les nouvelles localisations,
  - en supprimant automatiquement les localisations qui n'existent plus dans le fichier d'extraction,
  - en modifiant les localisations existantes par la prise en compte des modifications contenues dans le nouveau fichier d'extraction.
    En cas de modification, les paramètres positionnés par le développeur restent inchangés.

5.6. Test de l'application générée – Packaging

5.6.1. Test de l'application générée

5.6.1.1. Contrôle des versions

Si l'option du contrôle des versions détecte un déphasage, c'est que le Composant Applicatif et l'objet Proxy n'ont pas été générés avec le même numéro de version. Vous devez donc, selon le cas, :

- regénérer l'objet Proxy si vous avez regénéré uniquement le Composant Applicatif en modifiant sa version,
- ou mettre en exploitation dans VisualAge l'application graphique générée contenant le nouveau composant proxy si cela n'a pas été fait,
- ou mettre en exploitation le Composant Applicatif généré si cela n'a pas été fait.

Au sujet du contrôle des versions, voir également le sous-chapitre **1.2**, ***Compatibilité des Composants Applicatifs*** / Objets Proxy.

5.6.1.2. Middleware optimisé ou mode trace

Le middleware de Pacbench Client/Serveur est disponible en version optimisée ou en version trace. La dissociation des deux versions est matérialisée par l'extension des fichiers associés aux DLLs du middleware.

Les fichiers d'extension '001' correspondent aux DLLs utilisées en mode TRACE. Le mode TRACE permet d'obtenir un fichier de tous les événements, méthodes et objets manipulés par le middleware. Ce mode est activé à partir des deux variables d'environnement suivantes :

- **IXOTRACE** :

  Cette variable permet d'activer (**IXOTRACE=1**) ou de désactiver (**IXOTRACE=0**) la trace de l'API Middleware.

- **IXOTRACE_FILE** :

  Cette variable permet de spécifier, quand la trace est active (**IXOTRACE=1**), le chemin du fichier de trace.
  exemple : IXOTRACE_FILE=c:\tmp\ixo_err.txt

Le fichier des événements, méthodes et objets n'est jamais réinitialisé par les fonctions middleware. Il est donc conseillé de le détruire régulièrement, le middleware se chargeant de le recréer automatiquement.

Les fichiers d'extension '002' correspondent à la version optimisée des DLLs du middleware de Pacbench Client/Serveur. Ces DLLs ne contiennent pas le code permettant de tracer le fonctionnement des communications et n'interprètent donc pas les variables d'environnement réservées à cet effet.

A l'installation de Pacbench Client/Serveur sur la station de développement, les DLLs prêtes à être exécutées correspondent à celles de la version optimisée du middleware.

5.6.2. Packaging

Packager une applet ou une application permet de passer des phases de développement et de test à la phase d'exploitation, c'est-à-dire faire en sorte de pouvoir utiliser l'applet ou l'application en dehors de l'environnement de développement, en l'occurrence VisualAge. Dans VisualAge Java, cela consiste à exporter cette applet ou cette application.

5.6.2.1. Rappel : Prérequis

En phase d'exploitation, le fichier **VAPLOCAT.INI** doit se trouver dans le même répertoire que les applications finales. Lors de l'installation de ces applications, le développeur a la charge de s'en assurer.

- Applet

Pour la mise en exploitation d'une applet Java, les éléments suivants doivent être installés sur le poste de l'utilisateur :

- Un serveur HTTP
- Un browser Web Java 1.1 *enabled*
- La gateway et le relay sont installés sur la machine du serveur HTTP

- Application *standalone*

Pour la mise en exploitation d'une application *standalone* Java, le Java Runtime Environment (JRE) doit être installé sur le poste de l'utilisateur.

Pour plus de détails sur l'environnement d'exécution, reportez-vous au *Guide de l'Utilisateur Pacbench C/S, Vol. I – Concepts – Architectures – Environnements*. Voir également le sous-chapitre **5.5**, *Gestion de la communication.*

5.6.2.2. Export

5.6.2.2.1. Que faut-il exporter ?

Il est impératif d'exporter toutes les classes d'exécution utilisées par l'application fonctionnelle qui ne font pas partie des classes de base. Les classes d'édition telles que les classes **BeanInfo** ou les beans utilisés pour le maquettage rapide des propriétés de type Rubriques Pacbase sont facultatives.

Pour ce qui est d'une application Pacbench C/S, il faut donc exporter :

- tous les packages du projet contenant le runtime Pacbench C/S, sauf les packages **com.ibm.vap.beans** et/ou **com.ibm.vap.beans.swing** (selon le package utilisé dans votre application). Pour ces deux packages, n'exportez que les classes réellement utilisées.
- le projet contenant les composants Proxy générés,
- l'applet ou l'application elle-même, c'est-à-dire tout le projet dans lequel celle-ci se trouve ou alors le ou les package(s) qui la compose(nt). Dans notre exemple, il s'agit du package **vap.sample** (pour l'exemple V1) ou **example.swing** (pour l'exemple V2).
- éventuellement les beans externes : dans l'exemple V1, nous utilisons le bean **IMulticolumnListbox** ; il convient donc d'exporter tout le package qui le contient, soit le package **COM.ibm.ivj.javabeans**.

5.6.2.2.2. Mise en place

- **Pour une applet**

Avant de réaliser l'export d'une applet, vous devez créer un répertoire sur la racine du serveur HTTP, destiné à recevoir le résultat de l'export. Par exemple **c:\www\html\codebase**. Ce répertoire constitue la racine des classes Java dans le serveur HTTP.

☞ Vous pouvez exporter les fichiers **class** directement dans l'arborescence du serveur HTTP, soit dans **codebase** dans notre exemple, ou dans un autre répertoire. Dans ce dernier cas, il faudra copier ces fichiers dans ce répertoire avant la mise en exploitation, en veillant à respecter l'arborescence des packages.

- **Pour une application *standalone***

Dans ce cas, l'emplacement du répertoire destiné à recevoir le résultat de l'export importe peu. Il suffit de déclarer ce répertoire dans la variable **CLASSPATH**.

5.6.2.2.3. Comment exporter ?

Pour l'export proprement dit, réalisez les opération suivantes :

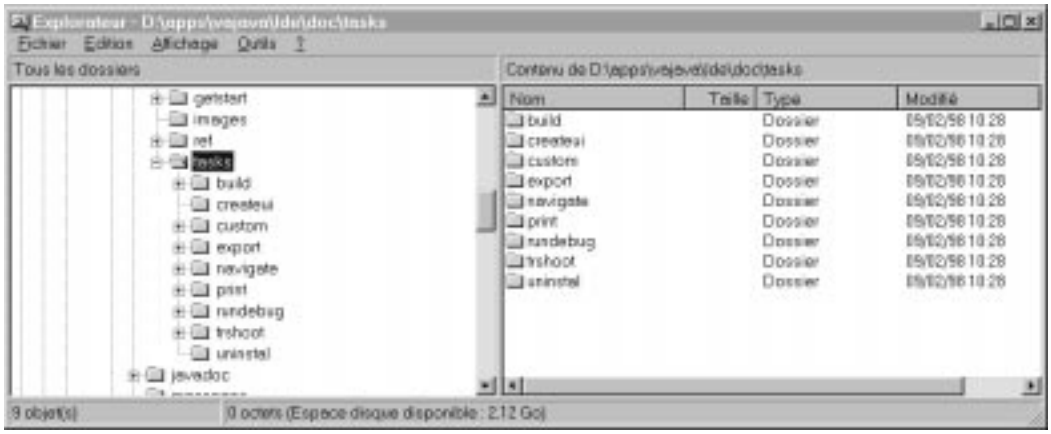
- dans le Workbench de VisualAge, sélectionnez tous les éléments à exporter,
- dans le menu **File**, sélectionnez le choix **Export**,
- dans la fenêtre **SmartGuide - Type of Export**, sélectionnez l'option **Class Files**, puis cliquez sur **Next**,
- dans la fenêtre **SmartGuide – Export to files**, entrez le nom du répertoire de sortie ou sélectionnez-le à l'aide du bouton **Browse**, en utilisant l'option **Create package subdirectories**,

Pour des informations complémentaires sur l'export, reportez-vous à la documentation en ligne VisualAge Java. Vous pouvez accéder à cette documentation depuis l'aide en ligne de VisualAge ou depuis l'Explorateur Windows 95 ou NT.

- Depuis VisualAge

  Dans le menu **Help**, choisissez **Tasks**. Le browser Internet s'ouvre alors : sélectionnez la rubrique *Exporting to the file system.*

- Depuis l'Explorateur Windows, ouvrez le répertoire **`Tasks`**, puis sélectionnez un fichier HTML dans le répertoire **`Export`**, le fichier **`overview.htm`** étant un point d'entrée suggéré.



5.6.2.2.4. Optimiser le temps de téléchargement des fichiers .class

Pour cette option, le JDK doit être installé dans l'environnement d'exécution.

Une fois l'export sous forme de fichiers **`.class`** effectué, vous pouvez optimiser le temps de téléchargement des classes lors de l'exécution en transformant tous ces fichiers en un seul fichier archive **`.jar`**.

Dans une fenêtre DOS ou OS/2, positionnez-vous dans le répertoire de sortie de l'export, puis entrez la commande :

```
jar cvf sample.jar com vap
```

où :

- **`sample.jar`** est le nom du fichier archive,
- **`com`** et **`vap`** représentent deux répertoires contenant tous les fichiers **`.class`** nécéssaires au fonctionnement de l'application finale.

Pour une applet, le fichier **`.jar`** obtenu doit être copié dans dans l'arborescence du serveur HTTP.

5.6.2.2.5. Ecriture d'un fichier HTML (applet seulement)

Enfin, l'écriture d'un fichier HTML contenant l'applet est nécessaire à son exécution dans un browser Web. Ce fichier permet de positionner certains paramètres, tels que la largeur et la hauteur de l'applet.

Pour cela, toujours dans le cadre de notre exemple, créez dans le répertoire **`c:\www\html\codebase\vap\sample`**, un fichier **`index.html`** contenant le texte suivant :

```
<HTML>
<TITLE>
Sample Applet
</TITLE>
<BODY>
<CENTER>
<APPLET code="vap.sample.SampleApplet.class"
WIDTH=1000" HEIGHT=1000
codebase="/codebase"></APPLET>
</CENTER>
</BODY>
</HTML>
```

ou le texte suivant, si vous avez constitué un fichier archive **`.jar`** :

```
<HTML>
<TITLE>
Sample Applet
</TITLE>
<BODY>
<CENTER>
<APPLET code="vap.sample.SampleApplet.class" WIDTH=1000
HEIGHT=1000
archive="/codebase/sample.jar"
codebase="/codebase"></APPLET>
</CENTER>
</BODY>
</HTML>
```

5.7. Déploiement de l'application

Pour déployer l'application, suivez les explications indiquées dans la documentation de VisualAge Java.

Mais vous devez aussi installer certains fichiers liés à l'utilisation de Pacbench C/S.

- **Pour une applet**

Le poste de l'utilisateur final doit être simplement équipé d'un navigateur.

- **Pour une application *standalone*** :
  - Si aucune gateway n'est utilisée, vous devez installer, sur le poste de l'utilisateur final :
    - **`MWJAVA.DLL`**,
    - les DLL du middleware,
    - **`VAPLOCAT.INI`**,
    - **`CHARCONV.TXT`** (conversion du code page),
    - **`VAPRUN.JAR`**,
    - **`VAPSWING.JAR`** si l'application utilise **`Swing`** ou **`VAPAWT.JAR`** si l'application utilise **`AWT`**.
  - Si une gateway est utilisée, vous devez installer, sur le poste où est installée la gateway :
    - **`GATEWAY.EXE`**,
    - les DLL du middleware,
    - **`VAPLOCAT.INI`**,
    - **`CHARCONV.TXT`** (conversion du code page).

    Dans ce cas, vous ne devez installer aucun de ces fichiers sur le poste de l'utilisateur final. En revanche, vous devez y installer le fichier :
    - **`VAPSWING.JAR`** si l'application utilise **`Swing`** ou
    - **`VAPAWT.JAR`** si l'application utilise **`AWT`**.

6. Développement d'un Client VisualAge Smalltalk

Après avoir généré les objets Proxy puis les avoir importés dans la station VisualAge, il ne vous reste plus qu'à les intégrer dans l'application graphique.

Après présentation des principes généraux, ce chapitre détaille pas à pas cette étape fondamentale : insertion des objets Proxy avec les liens de programmation impliquant les actions, attributs et événements, gestion des erreurs, gestion de la communication et enfin test de l'application.

Pour faciliter le développement et la maintenance des clients développés dans VisualAge, il est conseillé de faire coïncider une application VisualAge contenant des parts graphiques - Vues VisualAge ou fenêtres - avec une application fonctionnelle, bien que ce ne soit pas une obligation. L'application VisualAge est l'entité de packaging d'une application fonctionnelle.

Pour des informations sur le packaging, consultez la section **6.6.2**.

En suivant ces quelques conseils, vous éviterez l'apparition de certaines erreurs au moment de la Sauvegarde locale.

- Il est recommandé de développer votre application fonctionnelle dans une application VisualAge autre que celle dans laquelle vous avez généré vos objets Proxy.
- Avant de lancer la Sauvegarde locale, il est recommandé de vous assurer dans la Station VisualAge que le *manager*, *group member*, *owner* et *developer* correspond à un seul et même développeur pour l'application à sauvegarder.
- Si l'application à sauvegarder dans le Référentiel doit être exportée ultérieurement dans un autre environnement, il est recommandé de faire en sorte que le *manager*, *group member*, *owner* et *developer* soit le *library supervisor*.

Le développement d'un Client Smalltalk avec des Proxy Vues Logiques est déconseillé dans la présente version de Pacbench Client/Serveur car ses fonctionnalités sont limitées. Par conséquent, vous ne trouverez dans ce chapitre aucune information sur l'utilisation des PVL.

Si vous souhaitez néanmoins des détails sur les particularités de ce mode de développement, référez-vous au *Guide de l'Utilisateur Module Client/Serveur : Clients Graphiques 2.0*.

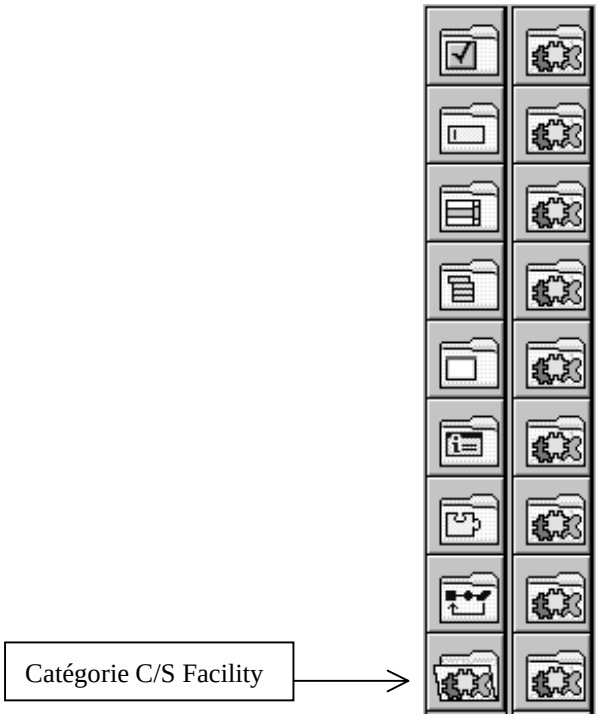
6.1. Principes généraux

☞ Si votre Proxy ne contient qu'une seule Proxy Elémentaire (nécessairement une Proxy Racine), les attributs, actions et événements associés aux lectures massives ou aux nœuds références ou dépendants ne sont pas disponibles.

6.1.1. Utilisation des objets Proxy dans le Composition Editor

Une fois importée dans la Station VisualAge, la Proxy Vue de Dossier est utilisable par un part graphique.

Pour placer la Proxy Vue de Dossier sur la Free Form Surface, sélectionnez dans la palette des parts la catégorie **`Client/Server Facility`**, le part correspondant à la PVD.



☞ Après sélection de la Proxy, un message vous demande si vous désirez l'insérer comme part variable ou constant.

Rappel: Dans VisualAge, un part constant représente un objet réel. Il ne peut être instancié qu'une fois. Un part variable est utilisé en remplacement de cet objet réel dans une autre Vue VisualAge : il permet donc la réutilisation de cet objet et son partage entre plusieurs Vues.

6.1.1.1. Option de dépliage dans le menu contextuel des Proxy Elémentaires

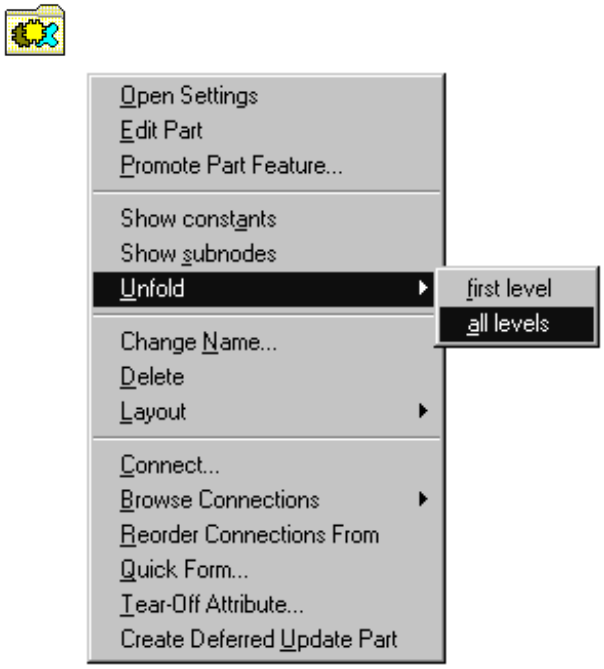
Maintenant que votre Proxy Vue de Dossier est sur la Free Form Surface, vous pouvez accéder à l'interface publique de la Proxy Racine. Cependant, l'interface publique des Proxy Elémentaires filles n'est pas encore accessible.

Pour la rendre accessible, vous devez d'abord les rendre visibles sur la Free Form Surface en utilisant l'option de dépliage des Proxy Elémentaires filles.

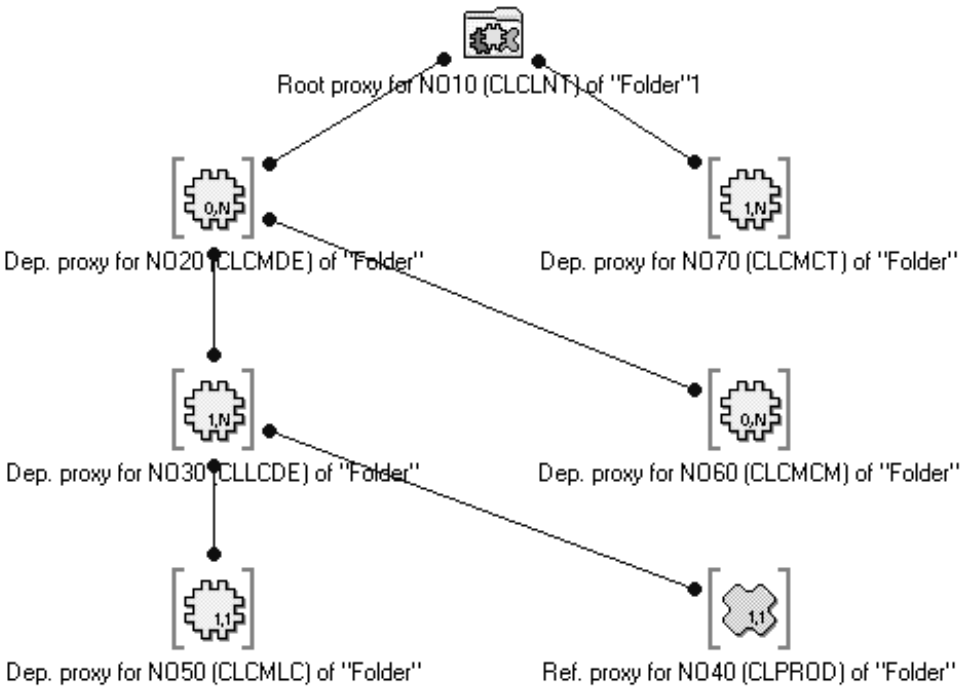
Pour cela, affichez le menu contextuel de la Proxy Racine et sélectionnez le sous-menu **`Unfold`** qui possède les deux choix suivants :

- Le choix **`first level`** permet d'afficher uniquement le niveau immédiatement inférieur.
- Le choix **`all levels`** permet de déplier tous les niveaux à la fois.

  Dans l'exemple ci-dessous, nous retrouvons la Proxy Vue de Dossier que nous avons générée et importée dans le chapitre précédent.



provoque :



Il est également possible d'afficher tous les niveaux, en utilisant **`first level`** plusieurs fois après le dépliage de chaque niveau.

Le sous-menu **Unfold** est en effet également disponible pour toute Proxy Elémentaire autre que la Racine, à condition qu'elle possède des Proxy filles.

Chaque Proxy Elémentaire de la Proxy Vue de Dossier est représentée par une icône spécifique qui dépend de son type ou de la cardinalité de la relation qui la lie à une Proxy Elémentaire parente.

Icônes	Types de Proxy Elémentaire possibles
	Proxy Racine
	Proxy Dépendante 0,N
	Proxy Dépendante 0,1
	Proxy Dépendante 1,N
	Proxy Dépendante 1,1
	Proxy Référence 0,1
	Proxy Référence 1,1

Chacune des 7 Proxy Elémentaires générées dans notre exemple correspond, côté serveur, à un noeud, c'est-à-dire à un Composant Applicatif élémentaire et la Vue Logique qu'il gère.

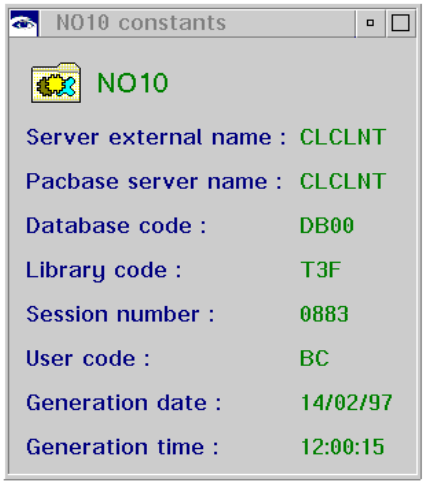
6.1.1.2. Contexte de génération

Ce contexte de génération est celui de la de la génération **GVC** de la Vue de Dossier associée.

Vous pouvez le consulter pour chaque Proxy Elémentaire visible sur la Free Form Surface .

Pour cela, affichez le menu contextuel de la Proxy Elémentaire et sélectionnez le choix **Show constants**.

La fenêtre suivante s'affiche :



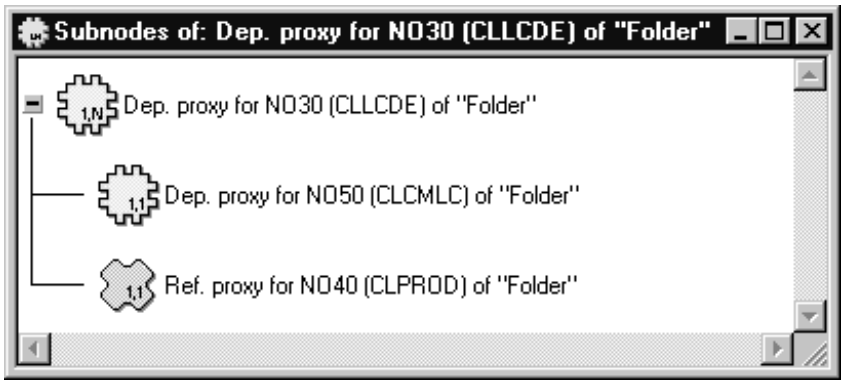
Si l'option de génération **`Optimize Generated Code`** a été sélectionnée, ces informations ne sont pas renseignées (elles sont remplacées par « ? ? ? »).

6.1.1.3. Visualisation des Proxy Elémentaires filles

Pour chaque Proxy Elémentaire visible sur la Free Form Surface, il est possible de demander l'affichage de ses Proxy filles sans passer par le sous-menu **`Unfold`** : leur dépliage est en effet inutile si vous n'avez pas besoin de les manipuler.

Pour cela, sélectionnez le choix **`Show subnodes`** dans le menu contextuel de la Proxy Elémentaire.

La fenêtre suivante s'ouvre :



La fenêtre affiche la ou les Proxy fille(s). Si l'une des Proxy filles contient également des filles, elle est précédée d'un signe + : cliquez dessus pour afficher ces dernières.

6.1.2. Utilisation des attributs

Un attribut correspond à une information gérée par un objet Proxy. Cette information définit une donnée élémentaire, une liste de données élémentaires ou une liste d'instances de données composées. Un attribut peut correspondre à une constante, à un paramètre ou à un résultat d'action. En fonction du contexte, il est initialisé par l'application graphique ou la Proxy.

On distingue deux types d'attributs :

- ceux qui représentent des variables technologiques. Ils permettent d'affiner le comportement des composants proxy dans VisualAge.
- ceux qui correspondent aux données de la Vue Logique.

La disponibilité d'un attribut est fonction du type de Proxy. Tous les attributs de l'interface publique sont documentés dans le *Manuel de Référence Clients Graphiques : Interface Publique des composants générés*.

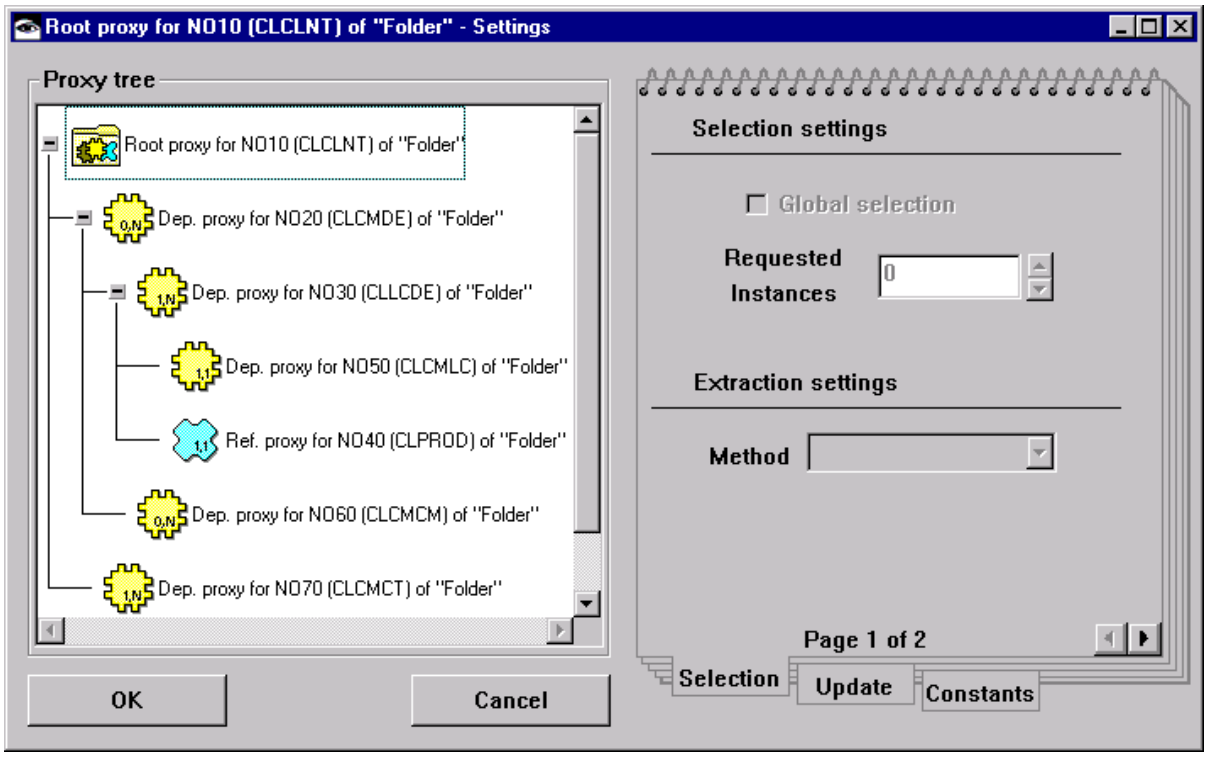
1.1.1.1. Paramétrage des attributs : Fenêtre Settings

Une fois la Proxy Vue de Dossier posée sur la Free Form Surface, avant d'utiliser les attributs associés aux objets Proxy, vous pouvez positionner les valeurs initiales de certains attributs technologiques via la fenêtre `Settings`. Ces valeurs initiales seront alors les valeurs par défaut de ces attributs, que vous pourrez toutefois modifier dynamiquement dans le cours du développement.

Ce paramétrage n'est pas obligatoire puisque un paramétrage par défaut est proposé, que vous pourrez également modifier dynamiquement par la suite.

Pour ouvrir la fenêtre `Settings`, sélectionnez le choix `Open Settings` du menu contextuel de la Proxy Racine ou double-cliquez sur l'icône correspondante.

La fenêtre `Settings` est disponible uniquement lorsque la Proxy Racine est utilisée comme part de type constant.



- **Sélectionner une Proxy**

La zone `Proxy Tree` présente, sous forme d'arborescence, la liste des objets Proxy qui constituent la Proxy Vue de Dossier. Elle permet de sélectionner la Proxy pour laquelle vous désirez définir les paramètres par défaut ou consulter ceux déjà positionnés.

☞ L'arborescence peut ne comprendre qu'une seule Proxy. Elle correspond nécesssairement à une Proxy de type racine et, à ce titre, elle en possède le comportement et les propriétés.

A l'ouverture de la fenêtre, seule la Proxy Racine est affichée. Le signe + précédant l'icône indique l'existence de Proxy filles. Cliquez sur + ou double-cliquez sur la Proxy parente pour les afficher. Pour afficher l'intégralité de l'arborescence, procédez de la même manière jusqu'à épuisement des signes +.

Pour ensuite ne conserver que les niveaux qui vous intéressent, cliquez sur le(s) signe(s) - ou double-cliquez sur la/les Proxy voulue(s).

Sélectionnez la Proxy qui vous intéresse.

La zone **`Selection Settings`** affiche alors les valeurs par défaut de cette Proxy.

- **Paramétrer les attributs technologiques d'une Proxy**

Les onglets **`Selection`**, **`Update`** du notebook permettent de spécifier des paramètres par défaut pour la Proxy sélectionnée dans la zone **`Proxy Tree`**. Chaque paramètre correspond à un attribut de Proxy.

- La première page de l'onglet **`Selection`** vous permet :
  - de spécifier ou consulter le nombre d'occurrences à lire. La zone **`Requested Instances`** correspond à l'attribut **`maximumNumberOfRequestedInstances`**. La valeur par défaut est celle spécifiée dans le serveur.
  - d'indiquer la méthode d'extraction souhaitée. La zone **`Method`** correspond à l'attribut **`extractMethodCode`**.
  - d'activer ou de désactiver l'indicateur de sélection globale des instances de la Proxy issues d'une action de lecture multi-instances. La zone **`Global Selection`** correspond à l'attribut **`globalSelectionIndicator`**.
  - d'activer ou de désactiver la mise en œuvre du tri local sur chaque nœud. Par défaut, le tri local est activé. (voir aussi l'attribut **`localSort`**).

  Les paramètres **`Global Selection`** et **`Requested Instances`** sont mutuellement exclusifs.
- La seconde page de l'onglet **`Selection`** vous permet d'indiquer la localisation du Dossier associé. Le paramètre **`Server location`** correspond à l'attribut **`location`**.
- L'onglet **`Update`** comporte trois options :
  - **`Refresh instances`** qui permet de réactualiser dans le client l'image locale des instances mises à jour par le serveur dans le cas où certaines Rubriques, comme les identifiants, sont calculées par le serveur. Cette option correspond à l'attribut **`refreshOption`**.
  - **`Check validity on server`** qui exécute un contrôle par intervalles et un contrôle utilisateur des Rubriques pour lesquelles ces options ont été positionnées dans le serveur. Cette option correspond à l'attribut **`serverCheckOption`**.
  - **`local rows sort`** qui permet de choisir le critère de tri des listes (attribut **`rows`**) : critère local ou serveur. Par défaut, le critère de tri local est spécifié. Cette option correspond à l'attribut **`localsort`**.

☞ Tous ces attributs sont documentés dans le *Manuel de Référence Clients Graphiques : Interface Publique des composants générés*.

- **Consulter le contexte de génération**

La fenêtre `Settings` comporte également l'onglet `Constants` permettant, pour la Proxy sélectionnée, de consulter le contexte de génération `GVC` de la PVD associée.

☞ Pour consulter le contexte de génération d'une Proxy lorsque la fenêtre `Settings` n'est pas disponible, utilisez le choix `Show constants` de son menu contextuel.

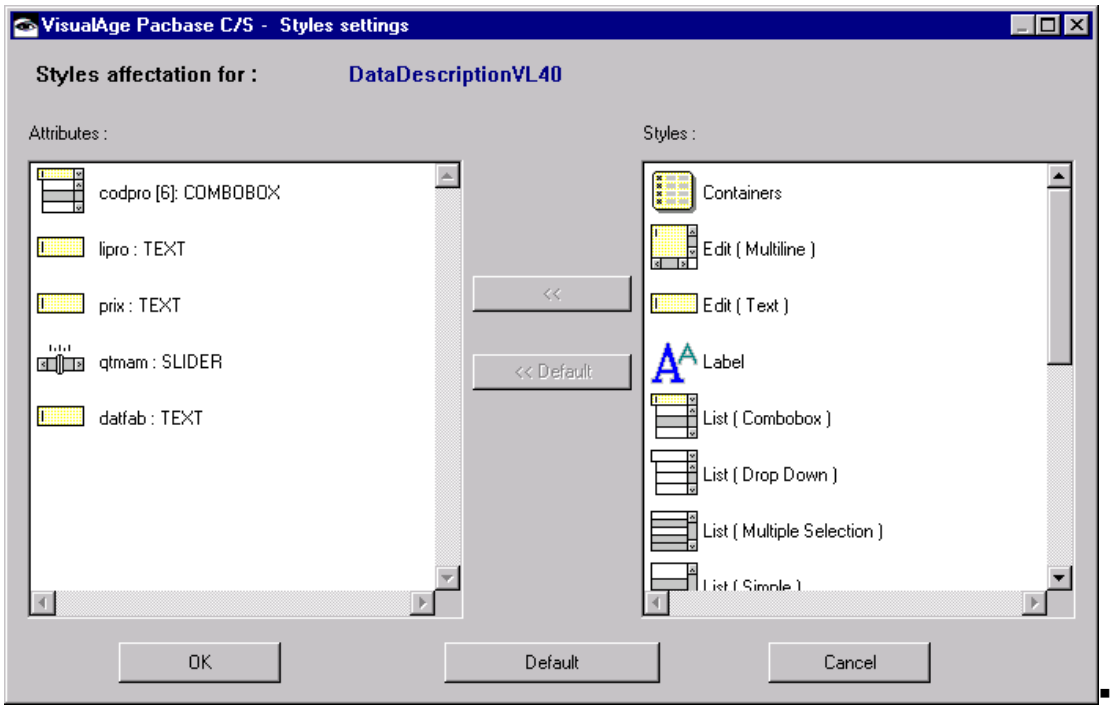
⚠ Si l'option de génération `Optimize Generated Code` a été sélectionnée, ces informations ne sont pas renseignées (elles sont remplacées par « ? ? ? »).

6.1.2.2. Définition d'une représentation graphique pour les Rubriques de l'attribut detail

Une représentation graphique par défaut - boîte saisissable, liste déroulante, etc - peut être définie dans le Référentiel pour chaque Rubrique de la Vue Logique. Le choix Quick-Form sur la Proxy permet de générer automatiquement cette représentation graphique dans la Station VisualAge.

Si vous souhaitez modifier cette présentation par défaut, vous devez exécuter les opérations suivantes :

- A partir du menu contextuel de la Proxy Elémentaire souhaitée, faites un Tear-Off Attribute sur son attribut `detail`.
- A partir du menu contextuel de `detail`, sélectionnez `Styles Settings`.



- Utilisez le bouton  pour appliquer à la ou aux Rubrique(s) sélectionnée(s) la présentation graphique souhaitée.
- Utilisez le bouton  pour restaurer la présentation définie dans le Référentiel pour la ou les Rubrique(s) sélectionnée(s).

- Utilisez le bouton Default pour restaurer, la présentation définie dans le Référentiel pour toutes les Rubriques de la Vue Logique associée à la Proxy.
- Une fois vos styles définis, cliquez sur **OK**. Ces styles seront générés automatiquement par la fonction Quick-Form.

6.1.2.3. Contrôle de longueur des champs de l'attribut detail

Pour chaque champ de l'attribut **detail**, lors des contrôles effectués pour une création ou une modification locale d'instance, la longueur de la valeur contenue dans l'attribut ne doit pas dépasser la longueur maximale de la valeur de cet attribut.

Le contrôle s'effectue systématiquement (sauf si l'attribut n'appartient pas au sous-schéma courant) même si l'attribut a été défini comme "non à contrôler dans le client".

Une erreur locale est envoyée en cas de longueur excessive.

☞ Il n'y a pas de contrôle de longueur sur les champs inclus dans les buffers utilisateur. Si la longueur est excessive, elle est tronquée à la longueur maximale.

6.1.2.4. Sélection du tri local ou serveur sur une liste d'instances

L'attribut **localSort** permet de spécifier si la Proxy trie les instances de l'attribut **rows** à partir du tri défini en local (**true**) ou laisse les instances triées dans l'ordre d'envoi du serveur (**false**).

Vous pouvez à tout moment modifier le type de tri.

☞ Vous pouvez aussi spécifier le type de tri en modifiant les paramètres locaux de la fenêtre **Settings**. Voir le paragraphe 6.1.2.1.

☞ Cet attribut n'est pas effectif dans les services utilisateur.

6.1.2.4.1. Tri local

Le tri local des instances de l'attribut **rows** correspond au fonctionnement standard de la Proxy si le paramétrage n'a pas été modifié après sa génération. Dans ce contexte, deux types de tri sont possibles :

- Si aucun critère de tri n'est défini en local, la Proxy trie implicitement les instances dans l'ordre croissant des identifiants définis sur la Vue Logique.
- Si un critère de tri est défini en local, la Proxy trie les instances dans l'ordre défini par ce dernier.

Dans tous les cas, la création locale d'une instance insère celle-ci en fonction du critère de tri courant appliqué dans l'attribut **rows**.

Le changement dynamique ou la suppression du critère de tri local induit immédiatement un tri sur les instances contenues dans l'attribut **rows**.

6.1.2.4.2. Tri serveur

Le tri serveur des instances de l'attribut **rows** est déclenché si l'attribut **localSort** est à **false** ou si le paramétrage standard du nœud concerné a été modifié. Dans ce contexte, les instances contenues dans l'attribut **rows** sont présentées dans l'ordre dans lequel elles ont été reçues du serveur, quel que soit le contenu du critère de tri défini en local.

Dans le contexte de gestion manuelle des collections ou de pagination en mode extend, les instances reçues sont ajoutées en fin de la collection existante dans l'attribut **rows**.

Toute instance créée localement est systématiquement ajoutée à la fin de la collection existante dans l'attribut **rows**. Dans ce contexte, une instance qui n'est pas positionnée en fin d'une collection qui est supprimée et recréée localement est transférée à la fin de la collection contenue dans l'attribut **rows**.

6.1.2.5. Utilisation spécifique du Quick-Form

6.1.2.5.1. Génération automatique d'une colonne par Rubrique de l'attribut rows

Un Quick-Form sur l'attribut **rows** d'une Proxy permet d'obtenir une table avec une colonne par Rubrique de la Vue Logique.

6.1.2.5.2. Liste des valeurs

Lorsqu'une représentation graphique est associée à la liste de valeurs d'une Rubrique dans le Référentiel et que cette représentation graphique est gérée par la fonction Quick-Form, le transfert du code correspondant à un libellé est géré automatiquement par la Proxy Racine ou la Proxy Dépendante.

6.1.2.6. Contrôles locaux

La Proxy Elémentaire exécute automatiquement les contrôles locaux lors de la création et de la modification d'une instance, via les actions **createInstance** et **modifyInstance**. Chaque Rubrique appartenant à la Vue Logique est contrôlée.

Les contrôles exécutés sont les suivants :

- Contrôles des listes de valeurs définies dans la description des Rubriques.
- Contrôles des intervalles définis dans la description des Rubriques.
- Contrôles de présence obligatoire définis au niveau de l'appel des Rubriques dans une Vue Logique. La présence des Rubriques de type identifiant ou de type foreign key pour une relation référence de cardinalité minimum de 1 est contrôlée automatiquement.

Si les contrôles locaux détectent une erreur, un message d'erreur est positionné via le Gestionnaire d'erreurs.

Ces contrôles peuvent être déclenchés de manière sélective pour chaque noeud de type racine ou dépendant du Dossier concerné. L'attribut permettant d'activer ou de désactiver le contrôle des Rubriques sur le serveur est **serverCheckOption**

Que l'on soit en émission ou réception de message, la détection d'une Rubrique vide est automatique.

En revanche, la Proxy n'assure pas les contrôles de numéricité et de date, qui sont pris en charge par les contrôles graphiques.

6.1.2.7. Gestion des sous-schémas

Les sous-schémas spécifiés dans la description de la Vue Logique peuvent être pris en compte lors des actions de sélection/lecture si les Composants Applicatifs gèrent la présence des Rubriques (options **VECTPRES=YES** ou **CHECKSER=YES**).

Chaque nœud possède deux attributs :

- **subSchema**, qui permet d'attribuer le sous-schéma désiré lors d'une action de sélection/lecture du Composant Applicatif du nœud. Cet attribut peut être alimenté à partir de l'attribut **subSchemaList**. Il est ignoré pour toute action de création ou de suppression.
- **subSchemaList**, qui permet de lister tous les sous-schémas disponibles pour le nœud. Comme il n'est pas possible dans le Référentiel VisualAge Pacbase d'affecter un nom au sous-schéma, chaque sous-schéma est désigné par **SubSchema<n>** (avec **n** de 01 à 10).

6.1.3. Utilisation des actions

6.1.3.1. Mise en œuvre

Une action correspond à un traitement qu'un objet Proxy peut exécuter. Elle est déclenchée à l'aide d'une connexion entre un événement de l'application graphique et le code d'une action d'une Proxy.

La disponibilité d'une action est fonction du type de Proxy. Toutes les actions de l'interface publique sont documentées dans le manuel *Manuel de Référence Clients Graphiques : Interface Publique des composants générés*.

6.1.3.2. Les différents types d'actions serveur

Les actions serveur exécutent des traitements implémentés dans un ou plusieurs Composants Applicatifs associés au Dossier. Ces actions émettent une requête vers les Composants Applicatifs qui renvoient un résultat sur la station de travail. Les requêtes et les réponses contiennent généralement des paramètres techniques, des instances de Vue Logique associées à un ou plusieurs noeuds et des informations contextuelles définies dans un buffer utilisateur.

Il faut distinguer deux types d'actions serveur.

- celles qui accèdent systématiquement au serveur :
  - **selectInstances**,
  - **readInstance**,
  - **readInstanceAndLock**,
  - **readInstanceWithFirstChildren**,
  - **readInstanceWithAllChildren**,
  - **readInstanceWithFirstChildrenAndLock**,
  - **readInstanceWithAllChildrenAndLock**,
  - **readAllChildrenFromCurrentInstance.**
  - **readAllChildrenFrom**
- celles qui ne font pas un accès systématique au serveur :
  - **readNextPage** : il y a accès au serveur sauf si lors de la précédente sélection, l'événement **noPageAfter** a été renvoyé.
  - **readPreviousPage** : il y a accès au serveur sauf si lors de la précédente sélection, l'événement **noPageBefore** a été renvoyé.
  - **readFirstChildrenFromCurrentInstance** : il y a accès au serveur, sauf si l'attribut **maximumNumberOfRequestedInstances** des Proxy Dépendantes est positionné à 0 et si l'attribut **globalSelectionIndicator** est positionné à false.

- **readFirstChildrenFrom** : il y a accès au serveur, sauf si l'attribut **maximumNumberOfRequestedInstances** des Proxy Dépendantes est positionné à 0 et si l'attribut **globalSelectionIndicator** est positionné à false.
- **checkExistenceOfDependentInstances** : il y a accès au serveur sauf s'il existe en local la possibilité de vérifier l'existence d'instances dépendantes.
- **updateFolder** : il n'y a d'accès au serveur que s'il existe au moins une instance du noeud concerné modifiée dans son attribut **updatedFolders**.

6.1.3.3. Gestion des lectures d'un Dossier

6.1.3.3.1. Lecture massive de la racine

La lecture massive de la racine d'un Dossier permet de lire sur un composant client toutes les occurrences du noeud racine du Dossier présentes dans la base de données. Les actions concernées sont **selectInstances** et **readNextPage**.

6.1.3.3.2. Lecture massive par anticipation des noeuds dépendants

Dans la cinématique des architectures client/serveur, une application graphique manipulant un Dossier cherche à acquérir des informations par anticipation pour minimiser les échanges avec les serveurs.

Dans un réseau hiérarchique, plusieurs actions d'anticipation des lectures peuvent être envisagées :

- La première action de type '**allChildren'** lit *toutes* les instances dépendantes de l'instance sélectionnée dans l'attribut **detail** de la Proxy Elémentaire parente.
- La deuxième action de type '**firstChildren'** ne lit que les instances *immédiatement* dépendantes de l'instance sélectionnée dans l'attribut **detail** de la Proxy Elémentaire parente.

La première action de lecture massive par anticipation n'est disponible que sur la Proxy Racine.

La deuxième action de lecture massive par anticipation est disponible sur la Proxy Racine ou sur les Proxy Dépendantes qui possèdent également des Proxy Dépendantes.

6.1.3.3.3. Transfert d'instance entre les attributs rows et detail

Le transfert d'instance entre l'attribut **rows** et **detail** permet d'alimenter l'attribut **detail** avec une instance initialement lue par une action de lecture massive.

Il correspond à une action de lecture locale qui alimente également toutes les instances locales des Proxy Dépendantes connues par la Proxy Vue de Dossier. Le transfert est réalisé par l'intermédiaire de l'action **getDetailFromDataDescription:**

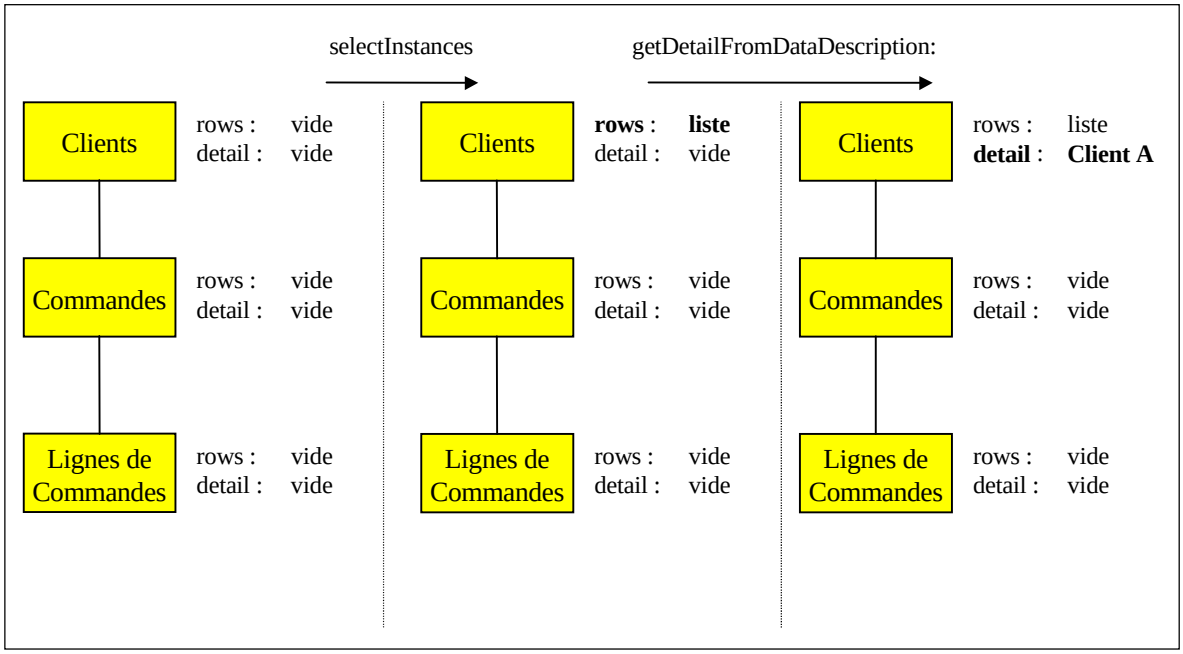
6.1.3.3.4. Lecture massive et transfert d'instance entre les attributs rows et detail : cinématique de fonctionnement

Cet exemple illustre l'alimentation de **`detail`** avec une instance préalablement transférée dans **`rows`** par une action de lecture massive d'un noeud racine ou dépendant et le principe de la lecture massive par anticipation des noeuds dépendants.

Il est basé sur la Proxy Vue de Dossier générée précédemment.

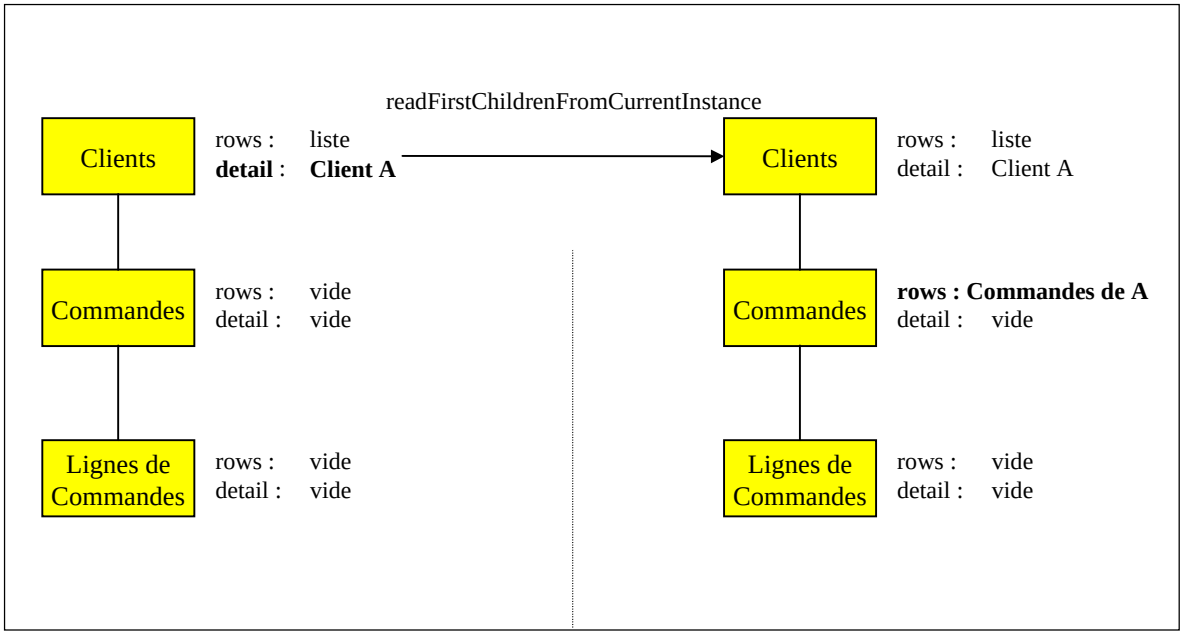
Pour les besoins de notre exemple, nous utilisons ici trois Proxy Elémentaires :

- la Proxy Racine **`NO10(CLCLNT)`** correspondant au noeud **`Clients`** qui gère les clients dans le système d'information décrit par le Dossier.
- la Proxy Dépendante **`NO20(CLCMDE)`** correspondant au noeud **`Commandes`** qui gère les commandes dans le système d'information décrit par le Dossier.
- la Proxy Dépendante **`NO30(CLLCDE)`** correspondant au noeud **`Lignes de Commandes`** qui gère les lignes de commandes dans le système d'information décrit par le Dossier.



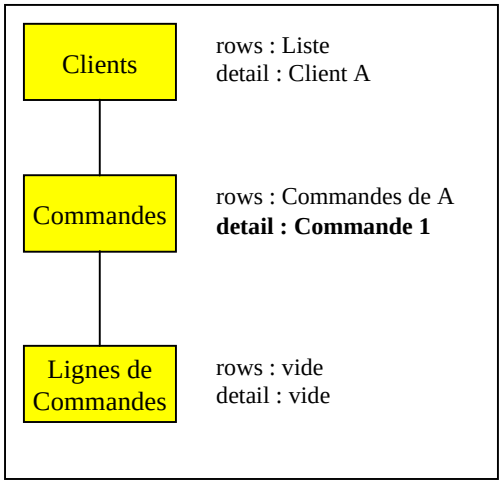
L'attribut **`detail`** du noeud Clients contient à présent le Client A. Ensuite, pour lire les instances dépendantes du Client A, c'est-à-dire ses commandes, vous disposez de trois solutions.

SOLUTION 1



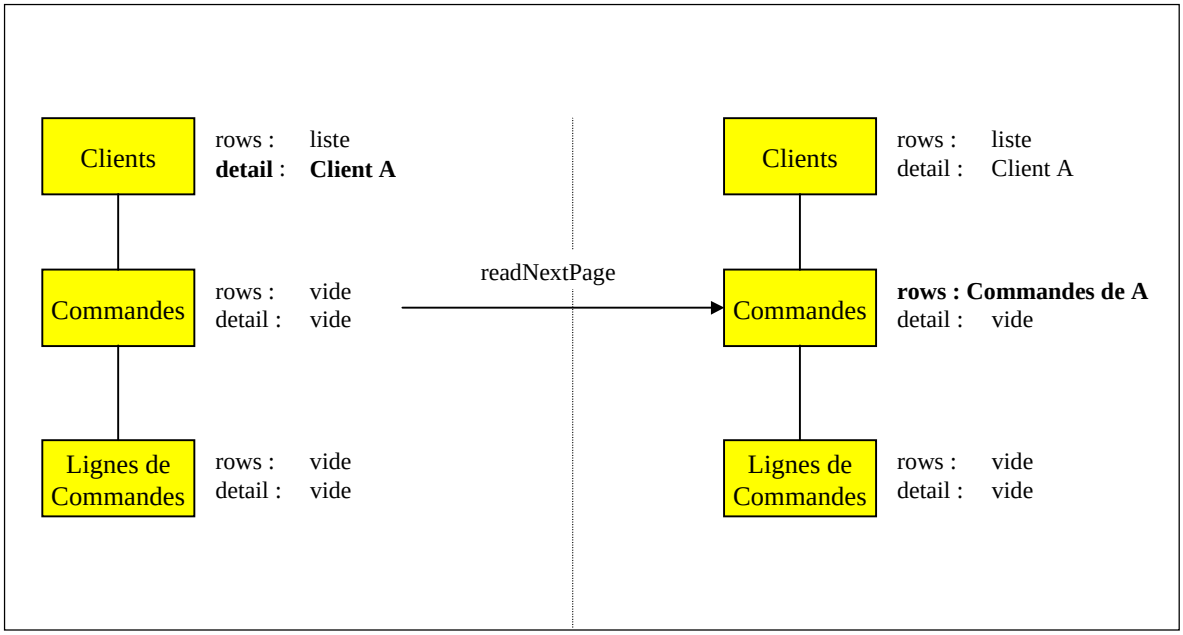
L'action **`readFirstChildrenFromCurrentInstance`** sur la Proxy Racine Clients alimente non seulement l'attribut **`rows`** de la Proxy Dépendante Commandes pour le Client A mais aussi l'attribut **`rows`** des éventuelles autres Proxy Elémentaires directement dépendantes de la Proxy Racine.

Dans ce contexte, l'action **`getDetailFromDataDescription:`** sur la Proxy Dépendante Commandes provoque :



Pour lire ensuite les lignes de commandes de la Commande 1 du Client A, utilisez l'action **`readFirstChildrenFromCurrentInstance`** sur Commandes ou l'action **`readNextPage`** sur Lignes de commandes.

SOLUTION 2

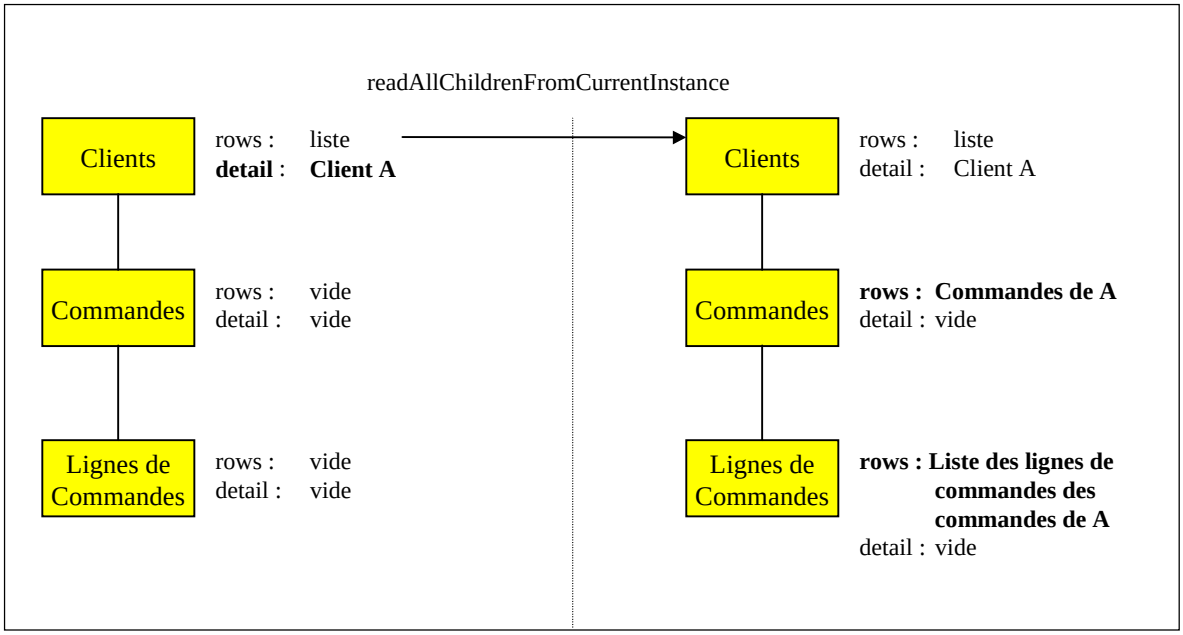


L'action **`readNextPage`** sur la Proxy Dépendante Commandes alimente son attribut **`rows`**. Les instances des éventuelles autres Proxy Elémentaires directement dépendantes de la Proxy Racine ne sont pas lues.

Dans ce contexte, l'action **`getDetailFromDataDescription:`** sur la Proxy Dépendante Commandes provoque le même résultat que dans la solution 1.

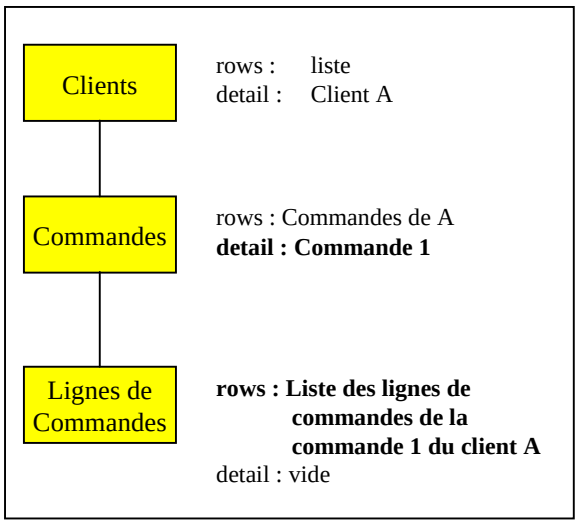
Pour lire ensuite les lignes de commandes de la Commande 1 du Client A, il faut procéder de la même manière que dans la solution précédente.

SOLUTION 3



L'action **`readAllChildrenFromCurrentInstance`** sur la Proxy Racine Clients lit non seulement toutes les commandes de A mais aussi toutes les éventuelles autres instances dépendantes de A quelque soit leur niveau hiérarchique : ainsi dans notre exemple, toutes les lignes de commandes de toutes les commandes de A sont lues.

Dans ce contexte, l'action **`getDetailFromDataDescription:`** sur la Proxy Dépendante Commandes provoque :



L'action **`getDetailFromDataDescription:`** alimente automatiquement l'attribut **`rows`** de la Proxy Dépendante Lignes de commandes avec les lignes de commandes de la Commande 1 du Client A, ces lignes ayant été transférées en local au préalable par l'action **`readAllChildrenFromCurrentInstance`**.

6.1.3.3.5. Lecture massive des noeuds références

La lecture d'un noeud référence est considérée comme une aide sur critères. Elle permet de montrer à l'utilisateur final une liste d'informations potentiellement référençables sur un noeud dépendant.

Les informations présentées à l'utilisateur correspondent aux informations nécessaires et suffisantes pour assister l'utilisateur final dans son choix.

Pour optimiser le volume des caractères transmis pour ce type de service, les Vues Logiques disposent d'un sous-schéma de type « aide sur critères » qui permet de sélectionner les Rubriques concernées au niveau du serveur.

Les lectures massives des noeuds références sont exécutées sur demande et ne peuvent pas être intégrées dans le traitement des lectures par anticipation. Les actions concernées sont **`selectInstances`** et **`readNextPage`**.

6.1.3.3.6. Principe de pagination dans les noeuds d'un Dossier

Deux types de pagination sont proposés sur les noeuds d'un Dossier :

- Le premier type de pagination, appelé pagination en mode *non-extend*, permet de paginer en avant et en arrière sur une collection prédéfinie, par l'intermédiaire d'actions spécifiques. Chaque action exécute une demande de lecture au serveur et son résultat remplace le résultat de la lecture précédente. Ce type de pagination n'est applicable que sur les noeuds racines ou références.
- Le deuxième type de pagination, appelé pagination en mode *extend*, permet de lire graduellement les instances d'une collection définie, au fur et à mesure des demandes de lectures des pages suivantes. Dans ce contexte, la lecture des pages précédentes disparaît et cette fonction est reprise en local par les ascenseurs du contrôle graphique qui présente la liste des instances. Ce type de pagination est applicable à tous les noeuds d'un Dossier.

6.1.3.3.7. Critères de sélection associés aux lectures massives

Les critères de sélection associés aux lectures massives sont des attributs élémentaires ou composés associés à chaque noeud d'un Dossier. Ils se répartissent en deux types :

- Les critères de sélection fonctionnels correspondent à l'identifiant et aux éléments nécessaires à la définition des méthodes d'extraction de la Vue Logique associée au noeud. Ces critères correspondent aux attributs suivants :
  - **`selectionCriteria`** définit les Rubriques de type identifiant et paramètres par valeur.
  - **`extractMethodCode`** définit le code de la méthode d'extraction souhaitée.
  - **`extractMethodList`** contient la liste des méthodes d'extraction disponibles.
- Les critères de sélection organiques correspondent aux informations permettant de maîtriser le volume des instances sélectionné sur chaque noeud.
  - **`globalSelectionIndicator`** est un attribut booléen qui, lorsqu'il est positionné à true, permet de lire la totalité des instances du noeud sur une requête de sélection.

- **maximumNumberOfRequestedInstances** est un attribut numérique qui indique, lorsque l'attribut **globalSelectionIndicator** est à false, le nombre d'instances d'un noeud à lire sur une requête de sélection. Cet attribut peut contenir la valeur 0. Dans ce cas, la branche n'est pas lue lors d'une lecture massive par anticipation.

6.1.3.3.8. Limitation du champ des lectures massives

Lors d'une lecture massive avec utilisation de critères de sélection, seules les instances qui correspondent à ces critères sont lues.

Cependant, dans le cas d'une lecture massive commandée par une lecture par anticipation de type **allChildren**, on considère que l'attribut **globalSelectionIndicator** est positionné à true sur chaque noeud.

6.1.3.3.9. Lecture d'une instance d'un noeud racine ou dépendant

La lecture d'une instance d'un noeud racine ou dépendant permet d'alimenter directement l'attribut **detail** d'un noeud sans passer au préalable par une sélection massive d'une collection des instances de ce noeud.

Ce type de lecture est considéré comme une sélection de collection et annule donc la sélection précédente même si celle-ci correspondait à une lecture massive. L'alimentation de l'attribut **detail** entraîne donc l'initialisation de l'attribut **rows**.

Les actions qui implémentent cette fonction de sélection permettent :

- Une lecture de l'instance sans ses dépendances (**readInstance**).
- Une lecture de l'instance avec ses dépendances de premier niveau (**readInstanceWithFirstChildren**).
- Une lecture de l'instance avec toutes ses dépendances (**readInstanceWithAllChildren**).

6.1.3.3.10. Lecture d'une instance d'un noeud référence

La lecture d'une instance d'un noeud référence ne peut pas activer de processus de lecture massive par anticipation. Elle permet de lire la totalité de la description de l'instance du noeud appelé dans son attribut **detail**.

Seule l'action **readInstance** est donc disponible sur un noeud référence.

Les noeuds références sont dépourvus d'action de mise à jour. Leurs attributs **rows** et **detail** sont indépendants. Le résultat de l'action **readInstance** n'initialise donc pas l'attribut **rows**.

L'attribut **rows** permet de visualiser les informations suffisantes pour permettre d'affecter une des instances du noeud référence à l'instance du noeud référençant.

L'attribut **detail** permet de consulter la totalité de la description d'une instance référencée.

La structure de ces deux attributs peut donc être différente.

6.1.3.3.11. Critères de sélection associés aux lectures d'une instance

L'identifiant de l'instance du noeud à lire est spécifié dans les critères de sélection fonctionnels associés au noeud.

Pour les noeuds dépendants, l'identifiant du noeud correspond à l'identifiant de la Vue Logique associée au noeud débarrassé des Rubriques définissant les identifiants des Vues Logiques hiérarchiquement supérieures. Ces identifiants hiérarchiques sont initialisés automatiquement par la Proxy Vue de Dossier en fonction de la navigation dans le Dossier.

6.1.3.4. Gestion des mises à jour d'un Dossier

6.1.3.4.1. Mises à jour locales

Les services de mise à jour locales sont disponibles sur chaque noeud de type racine ou dépendant du Dossier.

- Création d'une instance de noeud,
- Modification d'une instance de noeud,
- Annulation d'une instance de noeud.

Cependant, il existe d'autres règles inhérentes à la gestion des Dossiers :

- La création d'une instance d'un noeud dépendant n'est autorisée que si la hiérarchie des instances contenues dans les attributs **`detail`** des noeuds supérieurs existe.
- L'annulation d'une instance d'un noeud entraîne l'annulation récursive des instances locales des noeuds dépendants.

Pour permettre au développeur de gérer des messages permettant d'avertir l'utilisateur de l'impact d'une annulation en cascade, une action de vérification d'existence d'instances dépendantes est disponible sur les noeuds de type racine ou dépendant.

Cette action (**`checkExistenceOfDependentInstances`**) émet un résultat booléen true ou false. Si aucune dépendance n'est trouvée dans le cache local du Dossier et que l'instance concernée n'a pas été créée localement, elle émet une requête de vérification sur le serveur.

Pour qu'un utilisateur puisse revenir en arrière sur les manipulations locales d'une instance de Dossier, une action **`undoAllLocalFolderUpdates`** permet d'éliminer toutes les mises à jour locales sur tous les noeuds du Dossier appliquées depuis la dernière exécution d'une action de mise à jour serveur. Cette action n'est disponible que sur le noeud racine du Dossier. Une autre action **`undoLocalFolderUpdatesFor`** permet d'éliminer toutes les mises à jour associées à l'instance du nœud de Racine que vous aurez paramétré.

6.1.3.4.2. Mises à jour serveur

La Proxy Racine est seule à disposer des actions de mise à jour serveur.

Les mises à jour serveur correspondent aux actions permettant à un composant client d'envoyer au serveur l'ensemble des mises à jour locales effectuées depuis la dernière action de mise à jour de ce type.

Ces mises à jour concernent toutes les instances dépendantes modifiées. Elles peuvent concerner plusieurs instances de Dossier.

Lorsqu'une action de mise à jour serveur renvoie des erreurs, elle laisse le Dossier en statut « modifié localement » et seule la correction des erreurs et le renvoi des mises à jour, ou de l'action **`undoAllLocalFolderUpdates`** ou **`undoLocalFolderUpdatesFor`** permettent d'effectuer de nouvelles sélections de collection.

Les actions de mise à jour serveur exécutent, avant l'émission de la requête sur le serveur, un contrôle de l'intégrité du Dossier pour les instances créées localement. Cette action contrôle, pour chaque instance de noeud créé en local, les cardinalités minimum de chaque lien et émet une erreur si le nombre des instances dépendantes ne respecte pas les propriétés des liens associés.

Une action de mise à jour serveur peut être accompagnée d'une demande de sélection d'une nouvelle collection des instances du noeud racine. Cette demande de collection n'est exécutée que lorsque les mises à jour se sont effectuées sans erreurs. L'action correspondante est **`updateFolderAndSelectInstances`**.

Une action de mise à jour serveur peut être accompagnée d'une demande de rafraîchissement des instances mises à jour dans le cas où certaines Rubriques de ces instances, comme les identifiants, sont calculées par le serveur. Cette demande de rafraîchissement passe par l'utilisation de l'attribut **`refreshOption`**.

6.1.3.4.3. Gestion des mouvements utiles

La gestion des mouvements utiles, assurée par le cache local, est entièrement automatique et transparente pour le développeur.

Elle 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. Si plusieurs mises à jour locales ont été effectuées sur une même instance de noeud, seule la dernière sera envoyée au serveur.

6.1.3.4.4. Changement de sélection d'une collection

L'événement **`aboutToChangeSelection`** est envoyé au Client par la Proxy Racine lorsqu'une action de sélection serveur sur un noeud du Dossier remplace sa collection locale courante et que cette collection ou les collections dépendantes contiennent des modifications locales candidates à une mise à jour serveur. Si l'événement n'est pas intercepté lors de la programmation, les mises à jour sont perdues.

6.1.3.4.5. Réinitialisation des instances du cache local

L'action **`ResetCollection`** permet d'éliminer explicitement toutes les instances contenues dans le cache d'une Proxy Vue de Dossier avant de reconstituer une nouvelle collection. Cette action est disponible sur tous les types de nœuds d'une Proxy Vue de Dossier disposant de l'attribut **`Rows`**.

6.1.3.4.6. Gestion des collections d'instances

La gestion des collections d'instances peut se faire soit automatiquement, soit manuellement par le positionnement de l'attribut booléen **`ManualCollectionReset`** disponible sur tous les types de nœuds d'une Proxy Vue de Dossier. Le mode de gestion manuel permet de constituer des collections hétérogènes par une succession d'actions de sélection de collections et de pagination.

6.1.3.5. Actions asynchrones

6.1.3.5.1. Principes

La programmation asynchrone permet de dissocier l'action d'émission de la requête de celle qui permet de récupérer sa réponse. Vous pouvez utiliser ce type de programmation que vous utilisiez un protocole de communication asynchrone ou non.

Il est possible d’utiliser les composants Proxy en mode asynchrone, et ceci de façon totalement indépendante du middleware utilisé. Ainsi, on peut travailler en mode asynchrone au niveau Proxy, en utilisant une *location* dont le middleware est synchrone, et réciproquement.

Dans ce contexte, l'utilisateur final améliore sa productivité en soumettant à l'avance des requêtes dont il récupèrera la réponse au moment où il en aura besoin. Cette technique de travail par anticipation supprime la perception négative de l'inactivité associée aux temps de réponse.

De plus, les protocoles de communication permettent de sécuriser les requêtes ou les réponses dans des files de messages locales en garantissant l'acheminement du message quel que soit l'état du réseau.

Le mode de conversation d'une Proxy est défini par l'attribut **`setAsynchronous: aBoolean`**.

Lorsque la conversation est asynchrone, un événement **`asyncRequest`** est émis par la Proxy après chaque exécution d'une action de type serveur et l'identifiant de réponse permettant de récupérer ultérieurement la réponse associée à la requête est rendue dans l'attribut **`lastMessageId`**. La mémorisation de cet identifiant est à la charge de l'application.

Une réponse est explicitement récupérée par l'action **`getReplyOf:aReplyId`**.

Lorsque la réponse est disponible, le comportement de la Proxy est identique à celui déclenché par la réception d'une réponse associé à une action serveur synchrone.

Lorsque la réponse est indisponible, un événement **`replyPending`** est émis.

6.1.3.5.2. Actions globales ou associées à une instance

Certaines actions serveur, désignées comme actions globales, sont indépendantes de toute sélection, alors que d’autres dépendent d’une instance donnée contenue dans le cache local. En mode asynchrone, les actions globales stockent les identifiants de réponse dans une collection et chaque requête exécutée ajoute son identifiant dans cette collection ; les actions associée à une instance de Vue Logique stockent les identifiants de réponse dans une collection associée à l’instance concernée. Les collections d’identifiants de réponse associées aux actions globales sont perdues lorsque l’application qui utilise la Proxy est fermée. Une collection d’identifiants de réponse associée à une instance contenue dans le cache local est perdue lorsque cette instance est supprimée localement ou suite à un changement de collection.

- **Actions globales**

Les réponses associées aux actions suivantes peuvent toujours être traitées, indépendamment de la collection courante :

- `executeUserService`
- `readInstance` (RAC)
- `readInstanceWithFirstChildren` (RAC)
- `readInstanceWithAllChildren` (RAC)
- `readInstanceAndLock` (RAC)
- `readInstanceWithFirstChildrenAndLock` (RAC)
- `readInstanceWithAllChildrenAndLock` (RAC)
- `readNextPage` (RAC)
- `selectInstances`
- `lock`
- `unlock`
- `readPreviousPage`

- **Actions associées à une instance**

Les actions suivantes dépendent soit de l'instance `detail`, soit de l'instance passée en paramètre, soit de l'instance parente `detail` pour les nœuds dépendants :

- `checkExistenceOfDependentInstances`
- `readAllChildren(data)`
- `readFirstChildren(data)`
- `readAllChildrenFromCurrentInstance`
- `readAllChildrenFrom:`
- `readFirstChildrenFromCurrentInstance`
- `readFirstChildrenFrom:`
- `readInstance` (DEP)
- `readInstanceWithFirstChildren` (DEP)
- `readInstanceWithAllChildren` (DEP)
- `readInstanceAndLock ()` (DEP)
- `readInstanceWithFirstChildrenAndLock` (DEP)
- `readInstanceWithAllChildrenAndLock` (DEP)

6.1.3.6. Services utilisateur

Chaque noeud de type racine ou dépendant dispose pour mettre en oeuvre un service utilisateur des éléments suivants :

- Un attribut permettant d'avoir la liste des services utilisateurs disponibles sur ce noeud plus `nil`. Cet attribut est `userServiceList`.
- Un attribut permettant d'initialiser le code du service utilisateur à exécuter. Cet attribut est `userServiceCode`.
- Un attribut permettant de stocker localement des instances de Vue Logique à traiter pour le prochain service utilisateur. Cet attribut est `userServiceInputRows`.

- Un attribut permettant de présenter plusieurs instances de Vue Logique renvoyées par un service utilisateur. Cet attribut est **`userServiceOutputRows`**.
- Un attribut présentant les instances de Vue Logique candidates à l'exécution du prochain service utilisateur. Cet attribut est **`userDetail`**.
- Des actions locales permettant de mémoriser chaque instance de Vue Logique à envoyer au serveur pour l'exécution d'un service utilisateur. Ces actions sont **`create-UserServiceInstance`**, **`modifyUserServiceInstance`** et **`deleteUserServiceInstance`**.

Le noeud racine du Dossier dispose en plus des éléments suivants :

- Une action permettant d'exécuter l'ensemble des services utilisateur paramétrés sur chaque noeud du Dossier. Cette action est **`executeUserServices`**.
- Une action permettant d'effacer toutes les instances locales mémorisées pour tous les noeuds du Dossier. Cette action est **`resetUserServiceInputInstances`**.
- Une action permettant d'effacer l'instance courante mémorisée en local. Cette action est **`resetUserServiceCodes`**.

Ce principe permet d'exécuter dans la même requête un ensemble de 1 à n services utilisateur répartis sur des noeuds différents. L'ordre d'exécution de ces services correspond à l'ordre hiérarchique des noeuds, en parcourant l'arbre de haut en bas et de gauche à droite.

6.1.3.7. Verrouillage logique de la base

Les mécanismes d'upload-download associés à un Dossier augmentent le temps écoulé entre la lecture d'un Dossier et l'affichage de la mise à jour.

Dans ce contexte, sans mécanisme de verrouillage, deux utilisateurs peuvent modifier la même instance de Dossier. Les mises à jour cumulées deviennent difficiles à gérer.

Pour permettre à l'utilisateur d'utiliser un Dossier dans un mode d'appropriation exclusif, deux types de verrouillage d'une instance de noeud sont à sa disposition :

- Le verrouillage optimiste qui fonctionne sur le principe de la vérification de l'évolution d'un **`TimeStamp`** avant d'exécuter le traitement de mise à jour.
- Le verrouillage pessimiste qui s'approprie en mise à jour exclusive une entité au moyen de l'enregistrement d'une ressource spécifique. Dans ce cas, le traitement de mise à jour du Dossier est effectué avant de libérer la ressource exclusive.

Le traitement de verrouillage serveur est déclenché en fonction de l'exécution explicite d'une action spécifique disponible sur la Proxy Racine. Cette action est **`lock`**.

Le traitement de déverrouillage serveur est déclenché automatiquement avec l'exécution d'une action de mise à jour serveur ou explicitement en fonction de l'exécution d'une action spécifique disponible sur la Proxy Racine. Cette action spécifique est **`unLock`**.

L'écriture du traitement de verrouillage dans le Composant Applicatif racine est à la charge du développeur.

Ce traitement reçoit l'identifiant de l'instance de Vue Logique à verrouiller ainsi que le type de demande (verrouillage ou déverrouillage) à exécuter.

Il doit positionner en retour un statut permettant d'accorder ou de refuser le verrouillage ainsi que le **`TimeStamp`** ou le nom de la ressource utilisée.

En cas de refus de verrouillage, la Proxy Racine émet un événement **`lockFailed`** et toutes les actions de mise à jour locales ou serveur exécutées ultérieurement sont invalidées pour l'instance de Dossier spécifiée. Dans ce cas, le Dossier passe en statut « lecture uniquement ».

Le verrouillage logique est défini dans l'entité Dossier ou dans le Composant Applicatif en cas de développement mono-vue.

Lorsque l'option de verrouillage logique est active sur un Dossier, toute demande de lecture de l'attribut **`detail`** de la Proxy Racine peut être accompagné d'une demande de blocage logique au niveau du serveur.

6.1.3.8. Gestion de la présence des Rubriques

Les deux actions suivantes vous permettent de gérer la présence des Rubriques de la Vue Logique au niveau de la Proxy.

L'action **`isNull:<corub>`** permet de tester la présence ou l'absence de la Rubrique **`corub`**. Elle est générée pour toutes les classes **`DataDescription`** et **`UserDataDescription`** des nœuds dont le Composant Applicatif possède l'option **`NULLMNGT=YES`**.

L'action **`setNull:aBoolean on:<corub>`** permet d'indiquer la présence ou l'absence de la Rubrique avant toute action de mise à jour locale. Elle est générée pour toutes les classes **`DataDescription`** et **`UserDataDescription`** des nœuds dont le Composant Applicatif possède l'option **`NULLMNGT=YES`** et un service utilisateur ou un service de mise à jour.

Par défaut, toutes les Rubriques sont considérées présentes.

6.1.3.9. Gestion du contrôle des Rubriques

Vous pouvez gérer le contrôle des Rubriques de la Vue Logique au niveau de la Proxy avec l'action **`setCheck:aBoolean on:<corub>`**, qui permet d'activer ou d'inhiber les contrôles serveur sur une Rubrique avant toute action de mise à jour locale. Elle est générée pour toutes les classes **`DataDescription`** des nœuds racines ou dépendants dont le Composant Applicatif possède les options **`NULLMNGT=YES`** et **`CHECKSER=YES`** et un service de mise à jour.

Par défaut, toutes les Rubriques sont considérées à contrôler (si l'attribut **`serverCheckOption`** est positionné à **`true`**).

6.1.3.10. Gestion des sous-schémas

Toute action serveur de sélection et de lecture prend en compte le sous-schéma présent dans l'attribut **`subSchema`** et retourne donc les valeurs des Rubriques du sous-schéma. Si une action de sélection est suivie d'une action de pagination, le sous-schéma pris en compte est celui associé à l'action de sélection.

Toute action de création locale est effectuée hors sous-schéma.

Toute action de modification/suppression locale est effectuée sur le sous-schéma associé à l'instance, c'est à dire :

- si la modification/suppression est effectuée sur une instance créée localement, le sous-schéma est vide.

- si la modification/suppression est effectuée sur une instance lue, le sous-schéma est celui associé à la sélection de cette instance.

De plus, les trois actions suivantes sont spécifiques à la gestion des sous-schémas.

L'action **`resetSubSchema`** permet de réinitialiser l'attribut **`subSchema`** à vide, c'est à dire de ne sélectionner aucun sous-schéma.

L'action **`completeInstance`** permet de récupérer, par appel du Composant Applicatif associé à la Vue Logique, les valeurs des Rubriques n'appartenant pas au sous-schéma.

L'action **`belongsToSubschema`** permet de savoir si la Rubrique passée en paramètre appartient au sous-schéma associé à l'instance contenue dans l'attribut **`detail`**.

6.1.4. Utilisation des événements

Les événements émis par une Proxy permettent de déclencher des actions applicatives appartenant à l'application graphique. Ces traitements sont exécutés en connectant un événement de la Proxy à une ou plusieurs actions de l'application graphique. L'exécution conditionnelle des actions est facilitée par le fait qu'un événement est toujours accompagné de son événement contraire, les deux événements ne pouvant jamais être émis simultanément.

La disponibilité d'un événement est fonction du type de Proxy. Tous les événements de l'interface publique sont documentés dans le *Manuel de Référence Clients Graphiques : Interface Publique des composants générés*.

6.1.4.1. Gestion événementielle des lectures massives

La gestion événementielle des lectures massives permet au développeur d'avoir des informations sur l'état de la collection d'instances contenu dans un noeud. Chaque action de pagination disponible propose son propre système de pagination événementielle.

L'action de pagination en mode non-extend peut émettre les quatre événements suivants :

- **`noPageBefore`** : Cet événement est émis par un noeud racine ou référence à la fin de l'exécution d'une action de sélection de collection ou de pagination lorsque celle-ci ne renvoie pas d'erreur et que la page lue est la première de la collection courante.

- **`pageBefore`** : Cet événement est émis par un noeud racine ou référence à la fin de l'exécution d'une action de sélection de collection ou de pagination lorsque celle-ci ne renvoie pas d'erreur et que la page lue n'est pas la première de la collection courante.

- **`noPageAfter`** : Cet événement est émis par un noeud racine ou référence à la fin de l'exécution d'une action de sélection de collection ou de pagination lorsque celle-ci ne renvoie pas d'erreur et que la page lue est la dernière de la collection courante.

- **`pageAfter`** : Cet événement est émis par un noeud racine ou référence à la fin de l'exécution d'une action de sélection de collection ou de pagination lorsque celle-ci ne renvoie pas d'erreur et que la page lue n'est pas la dernière de la collection courante.

L'action de pagination en mode extend peut émettre les deux événements suivants :

- **`pageAfter`** : Cet événement est émis par tout type de noeud à la fin de l'exécution d'une action de sélection de collection ou de pagination avant lorsque celle-ci ne renvoie pas d'erreur et que le nombre d'instances contenu dans le noeud ne correspond pas au nombre d'instances total contenu dans la base.

- **`noPageAfter`** : Cet événement est émis par tout type de noeud à la fin de l'exécution d'une action de sélection de collection ou de pagination avant lorsque celle-ci ne renvoie pas d'erreur et que le nombre d'instances contenu dans le noeud correspond au nombre d'instances total contenu dans la base au moment de la requête.

La gestion événementielle est caractérisée par l'émission d'une information à un instant donné. Dans certains cas, il est nécessaire de connaître le statut d'un noeud après l'émission d'un événement de pagination.

Chaque noeud dispose donc d'une action qui permet de réémettre le dernier événement de pagination envoyé par ce noeud (**`getlastSelectResponseStatus`**). Les noeuds n'ayant pas participé à la dernière action de lecture émettent dans ce cas l'événement '**`notRead`**'.

6.1.4.2. Gestion événementielle des lectures d'une instance

Une Vue Logique peut être mappée sur une ou plusieurs entités de stockage physique. Dans ce contexte, la gestion événementielle de la lecture d'une instance de Vue Logique peut émettre l'événement suivant :

- **`notFound`** lorsque l'instance recherchée n'existe pas dans la base de données. Cet événement peut être émis lorsque la Vue Logique est mappée sur une ou plusieurs tables.

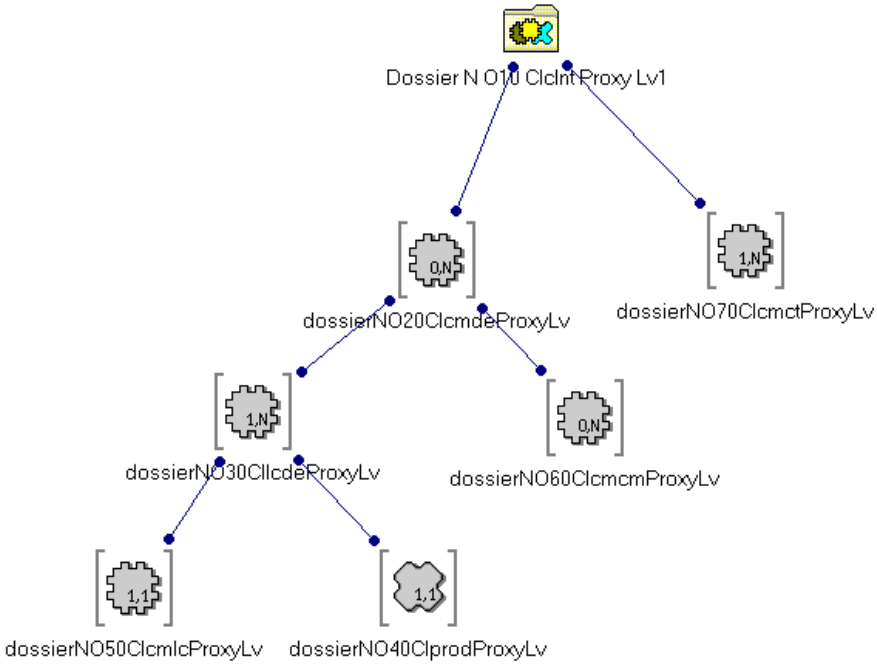
6.2. Exemple d'une application graphique standard

6.2.1. Introduction

Fourni à titre indicatif, cet exemple, sans prétendre à une description exhaustive de toutes les possibilités offertes, permet d'illustrer l'essentiel des principes de mise en oeuvre.

☞ Cet exemple est basé sur une Proxy Vue de Dossier contenant plusieurs Proxy Elémentaires. Mais les principes de mise en œuvre restent les mêmes pour une Proxy Vue de Dossier ne contenant qu'une seule Proxy Elémentaire. Si l'application ne contenait que la Proxy racine, seule la première fenêtre serait disponible ; les boutons permettant le débranchement sur les autres fenêtres seraient inactifs.

L'exemple est basé sur l'utilisation de la Proxy Vue de Dossier **`dossierNO10ClclntProxyLv`** générée dans le chapitre 3, ***Génération VisualAge Smalltalk***.



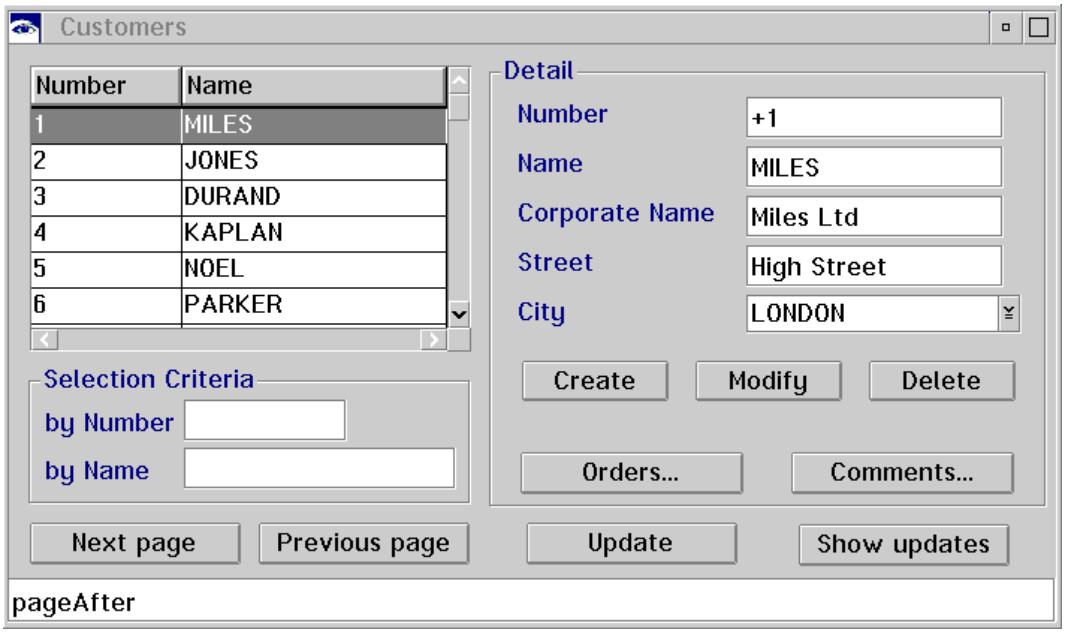
L'exemple ne manipule pas toutes les Proxy Elémentaires de la PVD mais seulement quatre d'entre elles :

- la Proxy Racine **`dossierNO10ClclntProxyLv`** correspondant au noeud **`Clients`** qui gère les clients dans le système d'information décrit par le Dossier.
- la Proxy Dépendante **`dossierNO20ClcmdeProxyLv`** correspondant au noeud **`Commandes`** qui gère les commandes dans le système d'information décrit par le Dossier.
- la Proxy Dépendante **`dossierNO30CllcdeProxyLv`** correspondant au noeud **`Lignes de Commandes`** qui gère les lignes de commandes dans le système d'information décrit par le Dossier.
- la Proxy Référence **`dossierNO40ClprodProxyLv`** correspondant au noeud **`Produits`** qui gère les produits susceptibles d'être commandés dans le système d'information décrit par le Dossier.

6.2.2. Présentation de l'application - exemple

L'interface graphique utilisateur développée comprend cinq fenêtres.

- **La fenêtre Customers**



Les fonctionnalités de cette fenêtre sont les suivantes :

- pagination avant et arrière pour sélectionner une liste de clients par plages de 50 : bouton **`Next Page`** et **`Previous Page`**.
- sélection d'une liste de clients par plages de 50 clients mais selon des critères :
  - affichage dans l'ordre croissant à partir d'un numéro de client donné. L'utilisateur doit saisir un identifiant dans la zone **`By Number`**, puis cliquer sur **`Next Page`** ou **`Previous Page`**.
  - ou affichage par ordre alphabétique à partir d'un nom de client donné. L'utilisateur doit saisir un nom dans la zone **`By Name`**, puis cliquer sur **`Next Page`** ou **`Previous Page`**.
- création, modification et suppression d'un client à partir de la fiche.

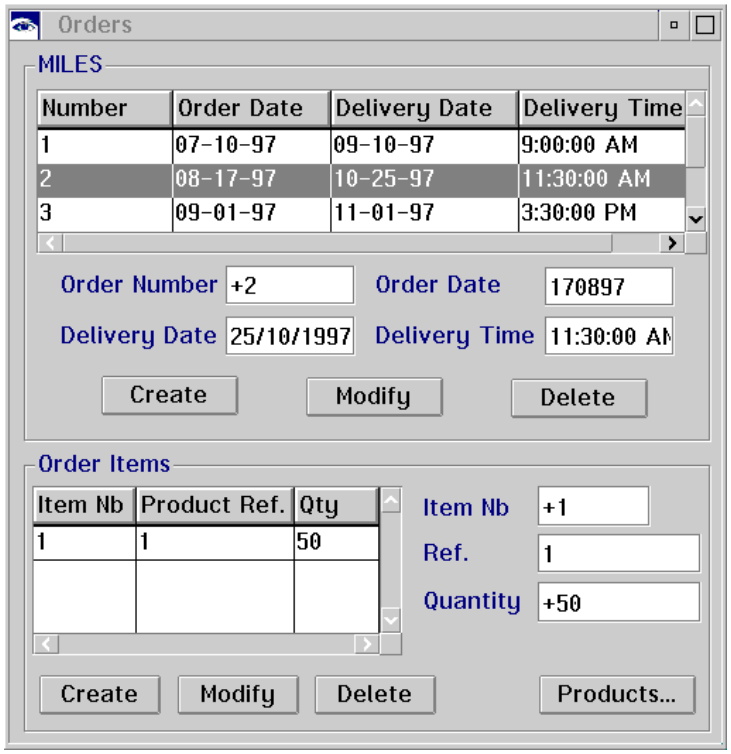
  Pour créer un client, l'utilisateur saisit les données associées au nouveau client dans la fiche, puis clique sur **`Create`**.

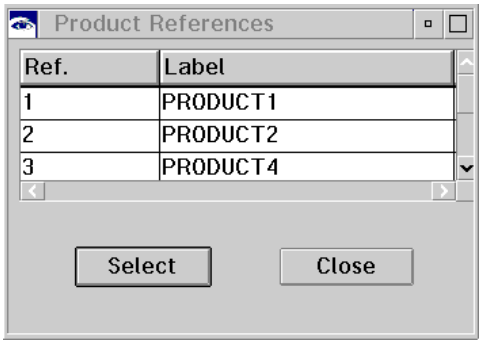
  Pour modifier un client, l'utilisateur sélectionne un client par double-clic sur une ligne de la liste. Les informations associées à ce client (identifiant, nom, nom de la société, adresse) s'affichent dans la zone **`Detail`**. Il doit alors cliquer sur **`Delete`**, ou **`Modify`** après la saisie des nouvelles données.
- navigation vers une fenêtre présentant la liste des clients impactés par des mises à jour locales : bouton **`Show updates`**.
- mise à jour de la base de données par la prise en compte de tous les clients impactés par des mises à jour locales : bouton **`Update`**.
- navigation vers la fenêtre de gestion des commandes : bouton **`Orders...`**.

- affichage de messages au retour du serveur après sélection d'une nouvelle plage de clients. Ces messages permettent par exemple de savoir si la plage de clients sélectionnée est la dernière.
- Le bouton **`Comments...`** n'est pas développé. Son existence indique seulement la possibilité de développer une fenêtre qui utiliserait la Proxy Dépendante **`dossierNO70ClcmctProxyLv`** correspondant au noeud Commentaires. Les principes mis en oeuvre pour le développement de cette fenêtre sont identiques à ceux utilisés pour la fenêtre **`Orders`**.

- **La fenêtre Orders et la fenêtre Product References**

Les fonctionnalités de ces deux fenêtres sont étroitement liées puisque la seconde se présente comme une aide sur critères pour la première.





La fenêtre **`Orders`** s'affiche lorsque l'utilisateur clique sur le bouton **`Orders...`**de la fenêtre **`Customers`**.

Pour le client sélectionné dans la fiche de la fenêtre **`Customers`**, cette fenêtre permet :

- de consulter la liste de ses commandes.

- de créer, modifier ou supprimer une commande à partir du détail de la commande.
- de consulter, pour la commande sélectionnée, la ou les lignes de commande : zone **Order Items**.
- de créer, modifier ou supprimer une ligne de commande à partir du détail de la commande.
- d'ouvrir la fenêtre qui présente la liste des produits : bouton **Products...**.

La fenêtre **Product References** s'ouvre lorsque l'utilisateur clique sur le bouton **Products...** de la fenêtre **Orders**.

Cette fenêtre représente une aide à la saisie lorsque l'utilisateur modifie une ligne de commande dans la zone **Order Items**. Elle contient une liste de produits que l'utilisateur ne peut modifier. Elle dispense l'utilisateur de saisir à la main le numéro de référence du produit associé à la ligne de commande.

Le principe d'utilisation de cette aide à la saisie est le suivant :

- Lors de la création d'une ligne de commande, l'utilisateur doit saisir dans le détail de la ligne de commande, le numéro de la nouvelle ligne de commande, la quantité désirée pour ce produit, puis ouvrir la fenêtre **Product References**. Dans le tableau des produits de cette fenêtre, il doit sélectionner le produit qui l'intéresse et cliquer sur **Select**. La valeur contenue dans la colonne **Ref** du tableau des produits est automatiquement transféré dans la zone **Ref**. du détail de la ligne de commande. Il suffit alors de cliquer sur **Create**.
- Lors de la modification d'une ligne de commande, l'utilisateur doit sélectionner, dans la liste de la zone **Order Items**, une ligne de commande. Le détail de la ligne de commande sélectionnée s'affiche.

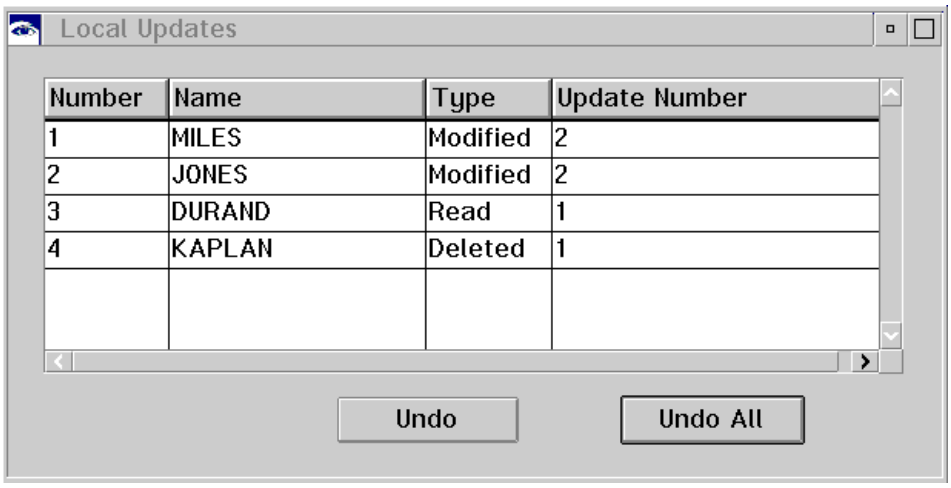
  Il faut ensuite ouvrir la fenêtre **Product References** et sélectionner dans le tableau des produits, celui par lequel on souhaite remplacer le produit courant, puis cliquer sur **Select** pour alimenter le détail de la ligne de commande avec le numéro de référence du nouveau produit. Après d'autres modifications éventuelles , l'utilisateur clique sur **Modify**.

Toutes les mises à jour, qu'elles aient été effectuées sur des clients, des commandes ou des lignes de commandes sont stockées en local. Pour les répercuter dans la base de données, l'utilisateur doit impérativement cliquer sur le bouton **Update** dans la fenêtre **Customers**.

- **La fenêtre Local Updates**

Cette fenêtre s'affiche lorsque l'utilisateur clique sur **Show Updates** dans la fenêtre **Customers**.

L'utilisateur y accède lorsqu'il souhaite consulter la liste des mises à jour effectuées depuis la dernière mise à jour de la base de données. Il a la possibilité de supprimer une ou plusieurs lignes dans cette liste et d'annuler ainsi pour les clients concernés les manipulations correspondantes avant une nouvelle mise à jour serveur.



Chaque ligne du tableau correspond à un client pour lequel une ou plusieurs mises à jour locales ont été effectuées.

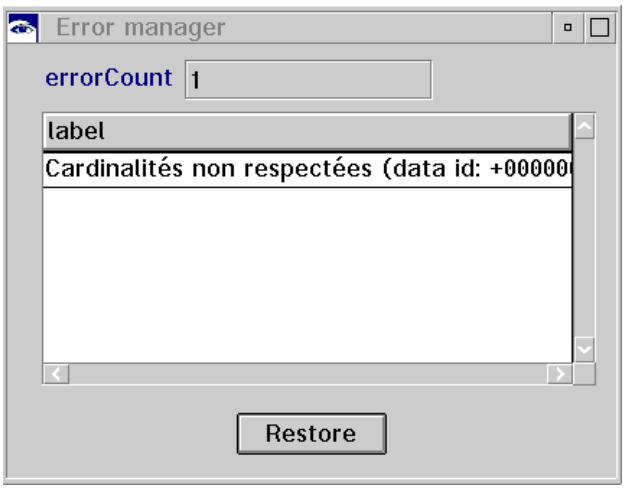
La colonne **`Type`** indique la nature de la mise à jour. Les valeurs possibles sont :

- **`Modified`** indiquant que des informations relatives à un client ont été modifiées,
- **`Deleted`** indiquant qu'un client a été supprimé,
- **`Created`** indiquant qu'un client a été créé,
- **`Read`** indiquant qu'une mise à jour (création, modification, suppression) a été effectuée sur une commande ou une ligne de commande du client.

L'utilisateur peut annuler toutes ces modifications en cliquant sur **`Undo All`** ou en supprimer une ou plusieurs qu'il a sélectionnées en cliquant sur **`Undo`**.

- **La fenêtre Error manager**

Cette fenêtre s'ouvre automatiquement si, suite à une action de l'utilisateur, le serveur, le gestionnaire d'échanges ou le cache local détecte une erreur.



La zone **`ErrorCount`** indique le nombre d'erreur détectées.

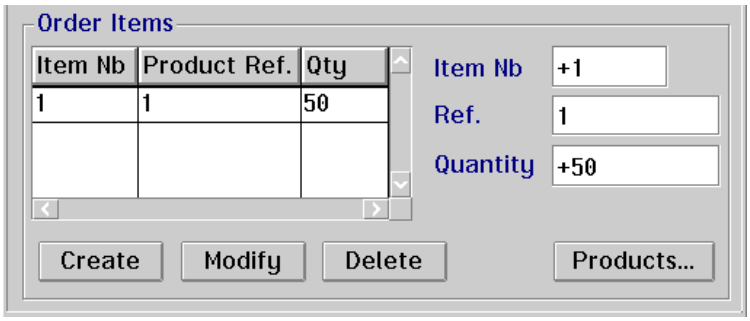
Pour certaines erreurs, l'utilisateur a la possibilité de restaurer automatiquement le contexte de l'erreur afin de connaître l'origine de l'erreur et d'y remédier avant une nouvelle tentative de mise à jour serveur.

Exemple Prenons un exemple concret pour illustrer le mécanisme de restauration automatique du contexte d'erreur.

Supposons que la LIGNE DE COMMANDE N°1 de la COMMANDE N°2 du CLIENT MILES soit à l'origine d'une erreur de nature à bloquer une demande de mise à jour serveur. La fenêtre des erreurs s'ouvre lorsque l'utilisateur clique sur le bouton **`Update`**. Pour l'assister dans sa recherche de la cause de l'erreur, l'utilisateur peut avoir recours à la fonctionnalité de restauration du contexte.

Pour cela il doit sélectionner la ligne correspondant à l'erreur dans liste et cliquer sur le bouton **`Restore`**.

Dans la fenêtre **`Orders`**, le détail de la LIGNE DE COMMANDE N°1 s'affiche automatiquement comme ci-dessous, à la place de la ligne de commande précédemment sélectionnée.



☞ Comme l'arbre de sélection est respecté, le détail de la COMMANDE N°2 et le détail du CLIENT MILES sont également affichées dans les fenêtres correspondantes, en remplacement des instances précédentes.

☞ La programmation de la fenêtre des erreurs est détaillée dans ce chapitre, sous-chapitre **6.4**.

6.2.3. Développement de l'application - exemple

Le développement de l'interface graphique utilisateur comprend la construction des quatre fenêtres qui composent l'application, dans l'ordre suivant : fenêtre **`Customers`**, **`Orders`**, **`Product References`**, **`Local Updates`**.

Dans la description de ces différentes étapes, l'utilisation des différents outils et fonctionnalités de VisualAge n'est pas détaillée. S'ils ne vous sont pas familiers, reportez-vous à la documentation appropriée.

Tout en gardant autant que possible une démarche séquentielle dans le développement, nous divisons, le cas échéant, la programmation des fenêtres en plusieurs parties selon des groupes cohérents de fonctionnalités.

A l'issue du traitement de chaque partie, nous présentons le Composition Editor tel qu'il doit apparaître en cachant éventuellement les liens précédemment développés pour mieux isoler ceux nouvellement créés.

☞ Il peut exister plusieurs solutions de développement pour une même fonctionnalité. Dans notre exemple, nous donnons la préférence à la solution visuelle.

6.2.3.1. Développement de la fenêtre Customers

6.2.3.1.1. Initialisation de la fenêtre et intégration de la PVD dans le Composition Editor

Les opérations à réaliser pour initialiser une fenêtre sont documentées dans la documentation VisualAge.

L'intégration de la Proxy Vue de Dossier dans le Composition Editor consiste à placer la Proxy Racine sur la Free Form Surface.

Sélectionnez dans la palette des parts la catégorie **`Client/Server Facility`**, puis le part correspondant à la Proxy.

Catégorie **`Client/Server facility`** : 

Part correspondant à la Proxy : 

6.2.3.1.2. Programmation des fonctionnalités liées à la liste : attribut rows

Cette étape de la programmation comprend les opérations suivantes :

- **Création de la liste des clients**

Pour créer la liste, il faut réaliser un Quick-Form sur l'attribut **`rows`** de la Proxy Racine. Nous obtenons un tableau (container) dont chaque colonne correspond à un attribut de la Proxy Racine.

Dans notre exemple, nous avons supprimé des colonnes pour ne garder que celles libellées **`Number`** et **`Name`**.

- **Programmation de la sélection**

Nous avons programmé deux types de sélection, avec possibilité de pagination avant et arrière :

- sélection d'une liste de clients par plages de 50 clients.
- sélection d'une liste de clients par plages de 50 clients mais selon des critères : affichage dans l'ordre croissant à partir d'un numéro de client donné ou affichage par ordre alphabétique à partir d'un nom de client donné.

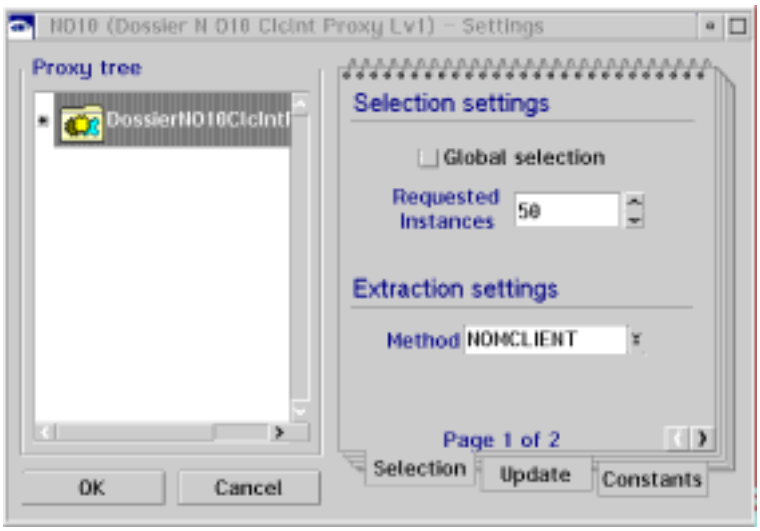
La possibilité de paginer en avant et en arrière est spécifiée dans le Dossier (ou dans le Composant Applicatif si aucun Dossier n'a été spécifié). Pour plus de précisions, voir le paragaphe **6.1.3.3**.

La programmation de ces services de sélection nécessitent les opérations décrites ci-dessous.

- Vous devez au préalable spécifier les critères de sélection associés aux actions de lecture : le nombre d'occurrences à lire et la méthode d'extraction à utiliser pour une sélection par nom; dans le serveur, une Rubrique de type identifiant constituant un paramètre d'extraction implicite, la méthode d'extraction correspondante est générée implicitement.

La solution la plus simple consiste à utiliser la fenêtre **`Settings`**.

La fenêtre est accessible via le choix **Open Settings** dans le menu contextuel de la Proxy Racine ou par double-clic sur le part correspondant.



- Vous devez ensuite faire un Quick-Form sur l'attribut **selectionCriteria** de la Proxy Racine. Vous obtenez deux zones de saisie correspondant aux deux critères de sélection accompagnées de leur libellé.
- Vous devez maintenant programmer les boutons **Next Page** et **Previous Page**.

  Pour cela, il faut connecter l'événement **clicked** des boutons correspondants aux actions **readNextPage** et **readPreviousPage** respectivement de la Proxy Racine.
- Il s'agit à présent de programmer le transfert d'instance entre la liste et la fiche de telle sorte que lorsque l'utilisateur double-clique sur une ligne de la liste, les informations relatives au client sélectionné s'affichent dans la zone **Detail**.

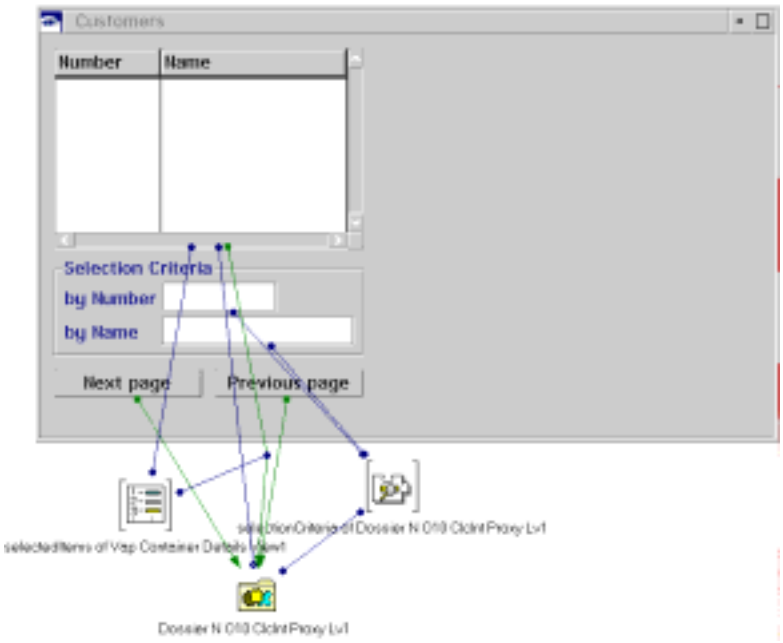
  Au préalable, il faut pouvoir extraire une instance unique à partir de la liste puisque l'attribut **rows** qui a permis de générer ce contrôle graphique est de type Ordered Collection, donc répété.

  Pour cela, la solution visuelle consiste à réaliser un Tear-Off de l'attribut **selectedItems** du container. Placez l'Ordered Collection ainsi obtenu sur la Free Form Surface. A ce stade, l'Ordered Collection est alimenté par plusieurs lignes de la liste.

  Il s'agit maintenant de transférer une instance donnée de la liste dans l'attribut **detail**. Vous devez d'abord connecter l'action **getDetailFromDataDescription:** de la Proxy Racine à l'événement **defaultActionRequested** du container. Le lien apparaît en pointillé, car l'action nécessite un paramètre : connectez l'attribut **first** de l'Ordered Collection au lien, via le paramètre **aDataDescription**.

- **Résultat dans le Composition Editor**

Une fois tous les traitements liés à la liste achevés, le Composition Editor doit se présenter comme ci-dessous :



6.2.3.1.3. Programmation des fonctionnalités liées à la fiche : attribut detail

La programmation des différentes fonctionnalités liées à la fiche est constituée des étapes suivantes :

- **Création de la fiche**

Au préalable, vous pouvez redéfinir la représentation graphique par défaut de certains attributs si celle spécifiée côté serveur pour les Rubriques correspondantes de la Vue Logique ne répond pas à vos besoins.

Pour cela, vous devez d'abord effectuer un Tear-Off de l'attribut **`detail`** de la Proxy Racine et placez le part ainsi obtenu sur la Free Form Surface.

Dans le menu contextuel de ce part, sélectionnez le choix **`Styles Settings`**.

Dans la fenêtre **`Styles Settings`**, affectez aux Rubriques souhaitées une nouvelle représentation graphique.

Dans notre exemple, une combobox a été affectée à l'attribut **`City`**.

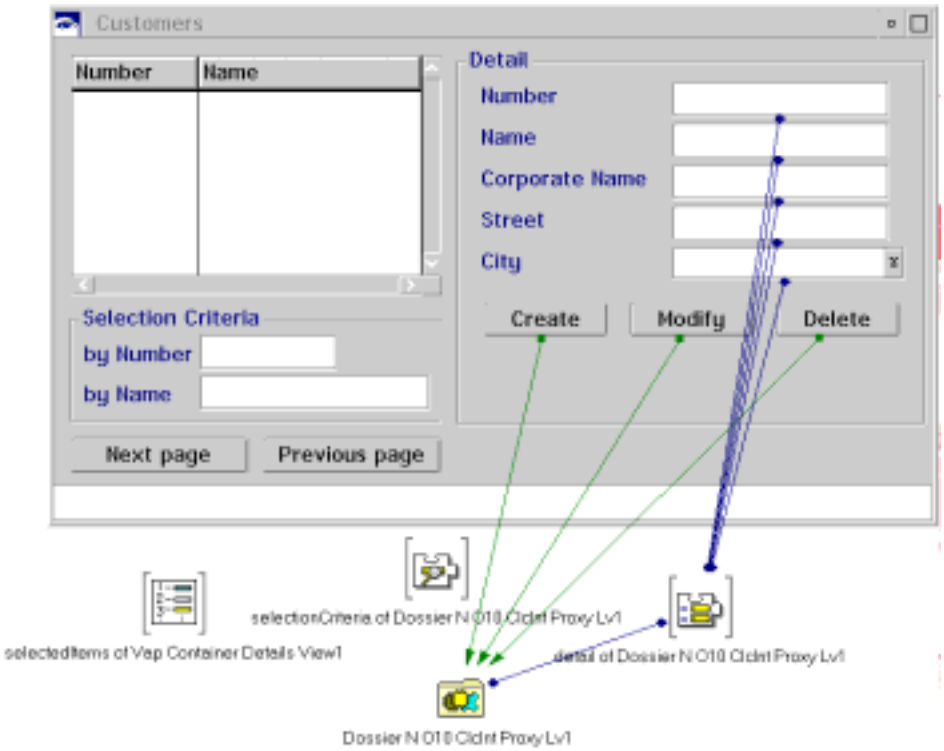
Maintenant, réaliser un Quick-Form sur le part **`detail`** de la Proxy Racine pour maquetter les champs correspondants aux différents attributs.

- **Programmation des services de mise à jour**

Pour programmer les services de mise à jour, vous devez connecter les boutons **`Create`**, **`Modify`** et **`Delete`**, via l'événement **`clicked`**, à la Proxy Racine via les actions **`createInstance`**, **`modifyInstance`** et **`deleteInstance`** respectivement.

- **Résultat dans le Compostion Editor**

A l'issue de cette étape, le Composition Editor doit se présenter comme ci-dessous :



6.2.3.1.4. Autres fonctionnalités

Les autres fonctionnalités de la fenêtre sont rappelées ci-dessous :

- mise à jour de la base de données par la prise en compte de tous les clients impactés par des mises à jour locales : bouton **`Update`**.
- affichage de messages au retour du serveur après sélection d'une nouvelle plage de clients. Ces messages permettent par exemple de savoir si la plage de clients sélectionnée est la dernière.
- navigation vers la fenêtre **`Local Updates`**.
- navigation vers la fenêtre **`Orders`**.

- **Programmation de la mise à jour serveur**

Vous devez connecter le bouton **`Update`**, via son événement **`clicked`**, à la Proxy Racine **`dossierNO10ClclntProxyLv`**, via l'action **`updateFolder`**.

- **Programmation de la barre de messages**

Pour programmer la barre la barre de messages, vous devez réaliser un Quick-Form de l'attribut **`accessInfoLabel`** de la Proxy Racine.

- **Programmation de la navigation vers la fenêtre Local Updates**

Cette étape est à réaliser une fois la fenêtre **`Local Updates`** construite. Nous supposons que tel est le cas et que, dans cette fenêtre secondaire, l'attribut **`self`** de la Proxy Racine variable a été promu. Le code de l'attribut promu est **`dossierNO10ClclntProxyLv1`**.

Vous êtes à présent de retour dans la fenêtre **Customers**. Il faut maintenant appeler le part correspondant à la fenêtre **Local Updates**. Pour cela, vous devez :

- Insérer un part de type View Wrapper et lui affecter un nom (dans l'exemple : **Updates**).
- Connecter l'attribut **self** de la Proxy Racine **dossierNO10ClclntProxyLv1** à l'attribut **dossierNO10ClclntProxyLv1** du View Wrapper.
- Connecter le bouton **Show Updates**, via l'événement **clicked**, au View Wrapper, via l'action **openWidget**.

- **Programmation de la navigation vers la fenêtre Orders**

Cette étape est à réaliser une fois la fenêtre **Orders** construite. Nous supposons que tel est le cas et que, dans cette fenêtre secondaire, l'attribut **self** de la Proxy Racine variable a été promu. Le code de l'attribut promu est **dossierNO10ClclntProxyLv1**.

Vous êtes à présent de retour dans la fenêtre **Customers**. Il faut maintenant appeler le part correspondant à la fenêtre **Orders**. Pour cela, vous devez :

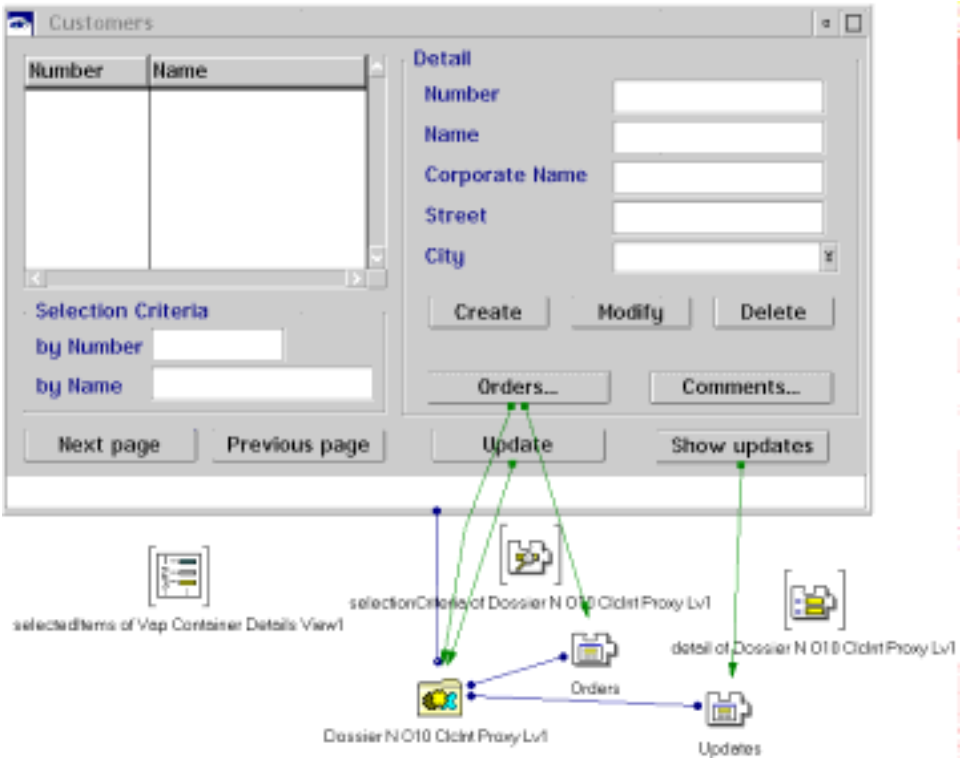
- Insérer un part de type View Wrapper et lui affecter un nom (dans l'exemple : **Orders**).
- Connecter l'attribut **self** de la Proxy Racine **dossierNO10ClclntProxyLv1** à l'attribut **dossierNO10ClclntProxyLv1** du View Wrapper.
- Connecter le bouton **Orders...**, via l'événement **clicked**, au View Wrapper, via l'action **openWidget**.

Dans notre exemple, nous avons également programmé le comportement suivant : le fait de cliquer sur le bouton **Orders...** déclenche la lecture sur le serveur des commandes du client courant et les lignes de commandes de chacune de ces commandes. A l'ouverture de la fenêtre les deux zones de liste de la fenêtre **Orders** seront donc alimentées.

Pour implémenter cette fonctionnalité, vous devez connecter le bouton **Orders...**, via l'événement **clicked**, à la Proxy Racine Clients, via l'action **readAllChildrenFromCurrentInstance**.

- **Résultat dans le Compostion Editor**

A l'issue de ces étapes, le Composition Editor doit se présenter comme ci-dessous :



6.2.3.2. Développement de la fenêtre Orders

La fenêtre **`Orders`** utilise deux noeuds : **`dossierNO20ClcmdeProxyLv`** correspondant au noeud Commandes et **`dossierNO30CllcdeProxyLv`** correspondant au noeud Lignes de commandes. Elle est développée dans une autre Vue VisualAge.

Nous ne présentons pas ici les fonctionnalités de cette fenêtre : veuillez vous reporter plus haut à la section **6.2.2**.

La fenêtre est divisée en deux parties : la partie supérieure manipule le noeud Commandes, la partie inférieure le noeud Lignes de commandes.

6.2.3.2.1. Intégration de la PVD dans le Composition Editor

Après avoir initialisé la fenêtre, insérez la PVD en tant que part variable sur la Free Form Surface. Il faut ensuite associer l'image de la PVD à ce part variable. Pour cela, vous devez promouvoir l'attribut **`self`** du part variable. C'est par l'intermédiaire de cet attribut promu que vous pourrez, une fois de retour dans la fenêtre **`Customers`**, connecter le part View Wrapper représentant la fenêtre **`Orders`** à la Proxy Racine **`dossierNO10ClclntProxyLv1`**, via son attribut **`self`**.

D'autre part, comme vous avez à manipuler l'interface publique des deux Proxy Elémentaires Dépendantes **dossierNO20ClcmdeProxyLv** et **dossierNO30CllcdeProxyLv**, vous devez utiliser l'option de dépliage pour les afficher. Dans notre exemple, nous déplions directement toute l'arborescence car la Proxy Référence **dossierNO40ClprodProxyLv** correspondant au noeud Produits est également utilisée par une seconde fenêtre appelée par la même Vue VisualAge.

6.2.3.2.2. Programmation des fonctionnalités liées au noeud Commandes

La programmation des fonctionnalités liées au noeud Commandes comprend les étapes suivantes :

- **Création d'une group-box**

Dans notre exemple, nous avons rassemblé dans une group-box tout ce qui relève de la gestion des commandes : tableau affichant la liste des commandes, zone affichant le détail d'une commande et boutons-poussoirs permettant de créer, modifier ou supprimer une commande. Le libellé de cette group-box est **Customer Name**. Pour l'utilisateur final, ce libellé est contextuel : il correspond au nom du client sélectionné dans le détail de la fenêtre **Customers**.

Après avoir positionné la group-box dans la fenêtre, pour implémenter ce comportement, vous devez :

- effectuer un Tear-Off de l'attribut **detail** de la Proxy Racine (noeud Clients) et poser le part variable obtenu sur la Free Form Surface.
- connecter le part variable, via l'attribut correspondant à la Rubrique Nom du Client de la Vue Logique à la group-box, via son attribut **label**.

- **Traitements liés à la liste**

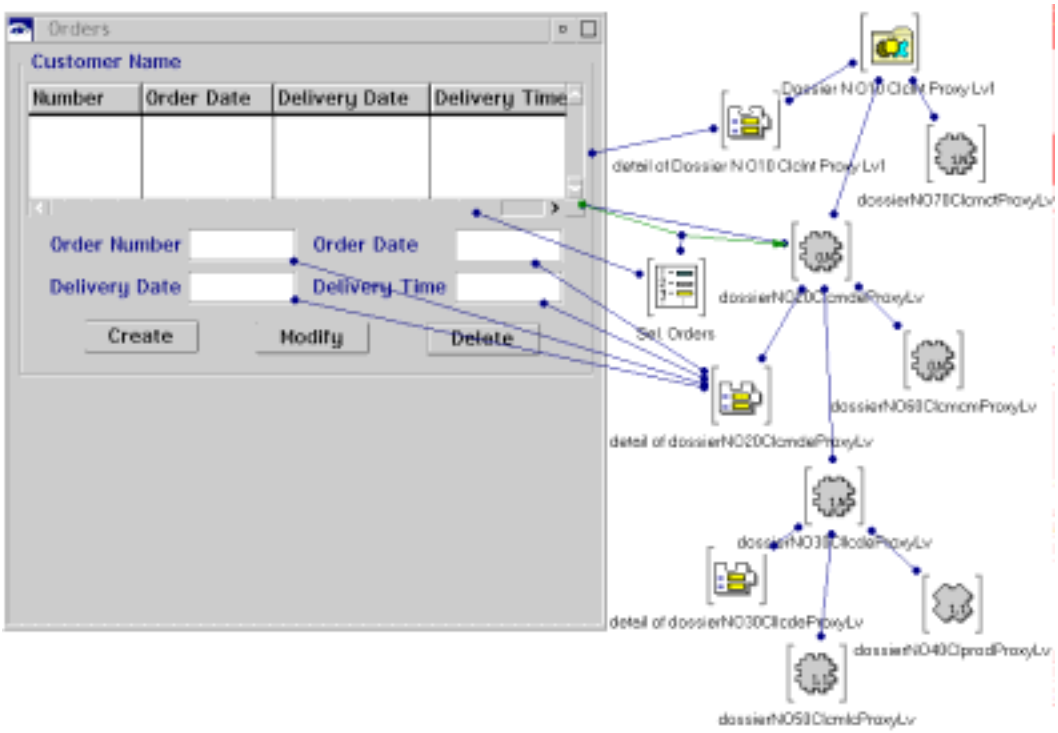
L'implémentation des traitements relatifs à la liste comprend plusieurs étapes.

- Réalisez un Quick-Form de l'attribut **rows** de la Proxy Dépendante **dossierNO20ClcmdeProxyLv**. Insérez le container obtenu dans la group-box.
- Vous devez ensuite programmer le transfert de l'instance sélectionnée dans la liste des commandes dans la zone correspondant au détail d'une commande. Les opérations nécessaires à la programmation du transfert étant identiques à celles requises pour le noeud Clients, veuillez vous reporter plus haut au paragraphe correspondant au point **6.2.3.1.2**.

- **Traitements liés à la fiche**
  - Réalisez un Quick-Form de l'attribut **detail** (le cas échéant après avoir réalisé un Tear-Off de l'attribut **detail** pour modifier la représentation graphique par défaut d'une ou plusieurs Rubriques via la fenêtre **Styles Settings**).
  - Pour programmer les services de mise à jour, vous devez connecter les boutons **Create**, **Modify** et **Delete**, via l'événement **clicked**, à la Proxy Dépendante via les actions **createInstance**, **modifyInstance** et **deleteInstance** respectivement.

- **Résultat dans le Composition Editor**

A l'issue de ces étapes, le Composition Editor doit se présenter comme ci-dessous :



Pour une meilleure lisibilité, les liens entre les boutons **Create**, **Modify** et **Delete** et la Proxy Dépendante **dossierNO20ClcmdeProxyLv** sont cachés.

6.2.3.2.3. Programmation des fonctionnalités liées au noeud Lignes de commandes

La programmation des fonctionnalités liées au noeud Lignes de commandes comprend les étapes suivantes :

- **Création d'une group-box**

Dans notre exemple, nous avons rassemblé dans une group-box ce qui relève de la gestion des lignes de commandes : tableau affichant la liste des lignes de commandes, zone affichant le détail d'une ligne de commande et boutons-poussoirs permettant de créer, modifier ou supprimer une ligne de commande. Le libellé de cette group-box est `Order Items`.

- **Traitements liés à la liste**

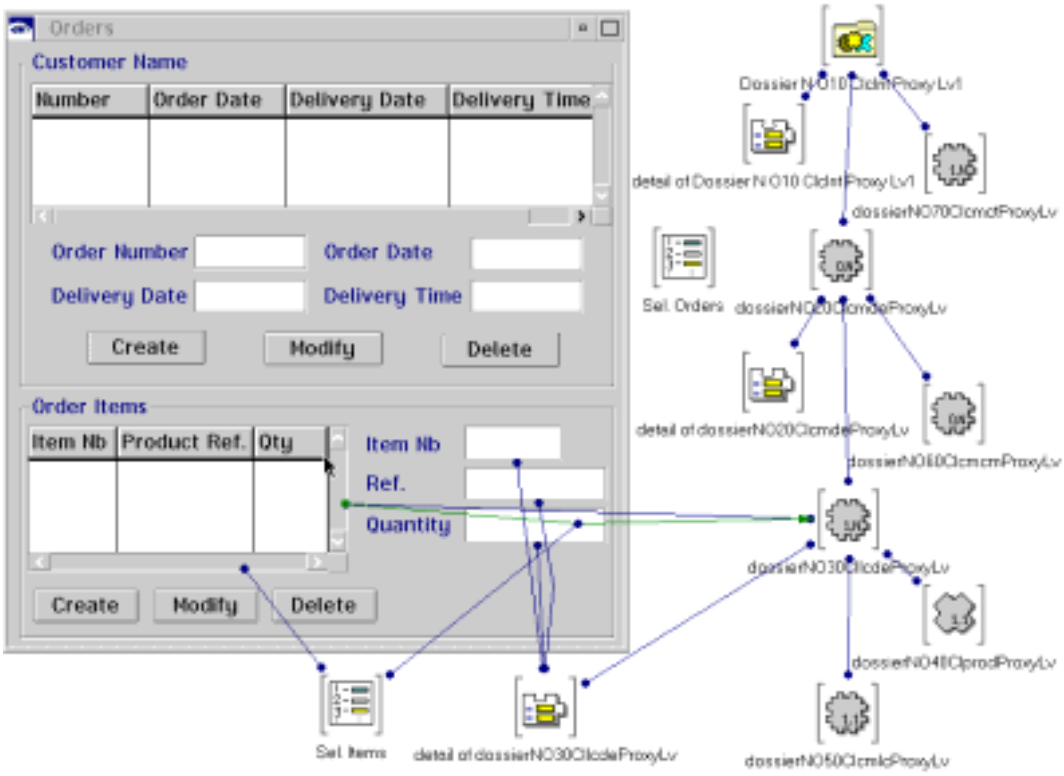
L'implémentation des traitements relatifs à la liste comprend plusieurs étapes.

  - Réalisez un Quick-Form de l'attribut `rows` de la Proxy Dépendante `dossierNO30CllcdeProxyLv`. Insérez le container obtenu dans la group-box.
  - Vous devez ensuite programmer le transfert de l'instance sélectionnée dans la liste des lignes de commandes dans la zone correspondant au détail d'une ligne de commande. A l'instar du noeud Commandes, les opérations nécessaires à la programmation du transfert étant identiques à celles requises pour le noeud Clients, veuillez vous reporter plus haut au paragraphe correspondant dans la Rubrique **6.2.3.1.2**.

- **Traitements liés à la fiche**
  - Réalisez un Quick-Form de l'attribut **`detail`** de la Proxy **`dossierNO30CllcdeProxyLv`** (le cas échéant après avoir réalisé un Tear-Off de ce même attribut pour modifier la représentation graphique par défaut d'une ou plusieurs Rubriques via la fenêtre **`Styles Settings`**).
  - Pour programmer les services de mise à jour, vous devez connecter les boutons **`Create`**, **`Modify`** et **`Delete`**, via l'événement **`clicked`**, à la Proxy Dépendante via les actions **`createInstance`**, **`modifyInstance`** et **`deleteInstance`** respectivement.

- **Résultat dans le Composition Editor**

A l'issue de ces étapes, le Composition Editor doit se présenter comme ci-dessous :



Pour une meilleure lisibilité, les liens entre les boutons **Create**, **Modify** et **Delete** et la Proxy Dépendante **dossierNO30CllcdeProxyLv** sont cachés.

6.2.3.3. Développement de la fenêtre Product References

La fenêtre **`Product References`** manipule la Proxy Référence **`dossierNO40ClprodProxyLv`** correspondant au noeud Produits. Cette fenêtre est développée dans la même Vue VisualAge que la fenêtre **`Orders`**.

- **Développement de la fenêtre**

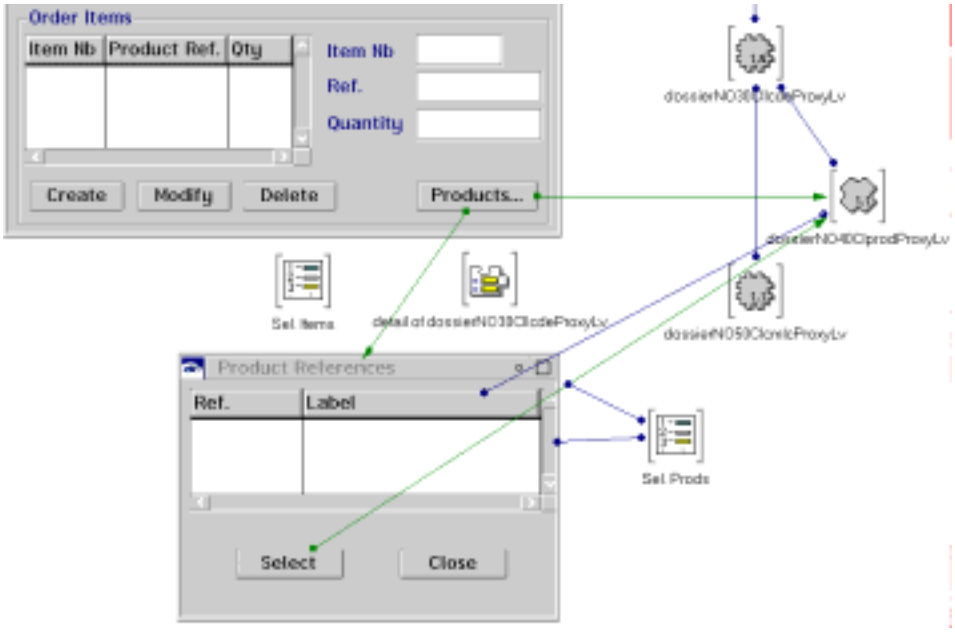
Pour programmer cette fenêtre, vous devez :

- Initialiser la fenêtre **`Product References`**.
- Connecter le bouton **`Products`**, via l'événement **`clicked`**, à la fenêtre **`Product References`**, via l'action **`openWidget`**.

- Connecter le bouton **`Products`**, via l'événement **`clicked`**, à la Proxy Référence **`dossierNO40ClprodProxyLv`**, via l'action **`selectInstances`**.
- Réaliser un Quick-Form de l'attribut **`rows`** de la Proxy Référence et placer le container ainsi obtenu dans la fenêtre.
- Réaliser un Tear-Off de l'attribut **`selectedItems`** du container et poser l'Ordered Collection obtenu sur la Free Form Surface.
- Vous devez maintenant programmer le transfert d'une seule ligne du tableau des produits vers le détail de la ligne de commande. Pour cela, connectez le bouton **`Select`**, via l'événement **`clicked`**, à la Proxy Référence, via l'action **`transferReferenceFromSelectedRow:`**. Connectez ensuite l'Ordered Collection, via l'attribut **`first`** au paramètre **`aDataDescription`** du lien.

- **Résultat dans le Composition Editor**

Le Composition Editor doit se présenter comme ci-dessous :



6.3. Exemple d’une application Web

6.3.1. Introduction

Une application Web dynamique développée avec Pacbench Client/Serveur utilise la même logique applicative qu'une application graphique standard. Avec Pacbench C/S, l'utilisation du Web Connection présente quelques particularités décrites dans ce sous-chapitre.

Les prérequis nécessaires à la réalisation d'une application Web sont documentés dans le *Guide Utilisateur Pacbench C/S, Vol. I : Concepts – Architectures - Environnements.*

Pacbench C/S fournit en outre sur le CD-ROM d'installation deux fichiers que vous pouvez installer avant de développer votre application Web. Ces extensions, utilisables pour un développement en mode dossier ou en mode simple, sont destinées à faciliter votre travail de programmation.

6.3.1.1. Conversion automatique des données d'un HtmlFormData

Les données saisies par un utilisateur final dans un navigateur sont toujours de type texte. L'extension **`cvfdata.dat`** permet à VisualAge de recevoir des données de type date ou entier déjà converties en utilisant les convertisseurs associés aux champs de saisie lors de la programmation de la page HTML. Pour charger ce fichier, effectuez les opérations suivantes à partir de l'Organizer :

- Positionnez-vous dans l'application **`AbtRunHtmlPageApp`**. Si elle n'est pas affichée, dans le menu **`Applications`** sélectionnez le choix **`View...`**, **`Show All Applications`**.
- Ensuite, dans le menu **`Parts`**, choisissez **`Import/Export`**, **`Import`**. Dans la boîte de dialogue qui s'ouvre, sélectionnez le fichier **`cvfdata.dat`**, puis cliquez sur **`OK`**.
- Chargez ensuite la dernière version des applications **`AbtRunHtmlPageApp`** et **`AbtEditHtmlPageApp`**.
- Si l'importation s'est déroulée correctement, la palette des parts de la catégorie **`Web Connection`** du Composition Editor contient le part supplémentaire suivant : **`HTML Converted Form Data`**.

6.3.1.2. Extension du Quick-Form HTML

L'extension **`qkhtml20.st`** permet d'obtenir un Quick-Form HTML avec un comportement identique au Quick-Form standard de Pacbench C/S. De plus, il permet de créer automatiquement une table HTML (ensemble de tags permettant une présentation des données sous forme de cellules de taille identique) contenant les libellés et les champs de saisie des attributs d'un composant proxy. Cette extension permet une présentation élégante des informations sans la nécessité de générer les tags HTML.

☞ Le fichier **`qkhtml20.st`** inclut le fichier **`R2VA.ALL`**, qui est l'extension du Quick-Form HTML de VisualAge.

L'extension **`qkhtml20.st`** permet de générer une représentation graphique des Rubriques d'une Vue Logique sous la forme d'une fiche.

Pour charger ce fichier, effectuez les opérations suivantes à partir du System Transcript :

- Dans le menu **`File`** sélectionnez le choix **`Open...`**. Sélectionnez le fichier **`qkhtml20.st`**. Un WorkSpace s'ouvre alors avec le contenu du fichier.
- Sélectionnez tout le contenu du fichier, puis dans le menu **`Edit`**, sélectionnez **`File In`**. Le code est alors chargé.
- Si le chargement s'est bien passé, le choix **`self as HTML table`** apparaît dans le sous-menu contextuel - qui s'ouvre lorsque vous sélectionnez le choix **`Quick HTML`** - de l'attribut **`detail`** d'une Proxy. Le choix **`self as HTML table`** génère les champs de mise à jour des attributs de la Proxy ainsi que les tags créant une table HTML.

6.3.2. Gestion d'un contexte applicatif

En ce qui concerne Pacbench Client/Serveur, si l'application que vous souhaitez réaliser comporte plusieurs fenêtres, vous devez gérer un contexte applicatif dans lequel la même Proxy est utilisée durant un même dialogue avec un utilisateur.

Cela implique qu'un part **`CGI Link Session Data`** doit être utilisé dans chaque page faisant appel à ce composant proxy. Dans la fenêtre **`Settings`** du part **`CGI Link Session Data`**, vous devez indiquer dans la zone **`Data type of the value`** le nom du composant proxy utilisé. Un Tear-Off de l'attribut **`value`** du **`CGI Link Session Data`** permet alors d'obtenir une instance variable du composant proxy.

Pour des informations complémentaires sur le part **`CGI Link Session Data`**, reportez-vous au Manuel *VisualAge Web Connection User's Guide*.

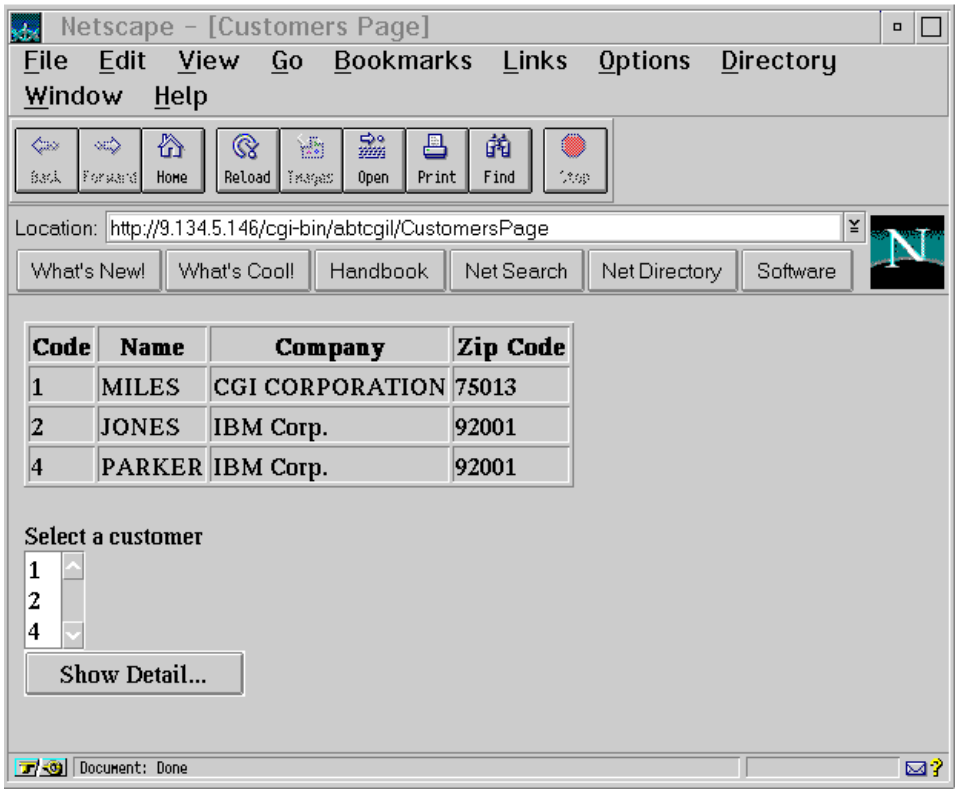
6.3.3. Exemple : Présentation de l'interface utilisateur

Cette application est basée sur la même Proxy Vue de Dossier que celle utilisée pour le développement de l'application graphique standard.

L'exemple manipule une seule Proxy Elémentaire, la Proxy Racine **`dossierNO10clclntProxyLv`**, correspondant au couple Composant Applicatif/Vue Logique qui gère les clients dans le système d'information de l'entreprise.

L'application contient deux pages.

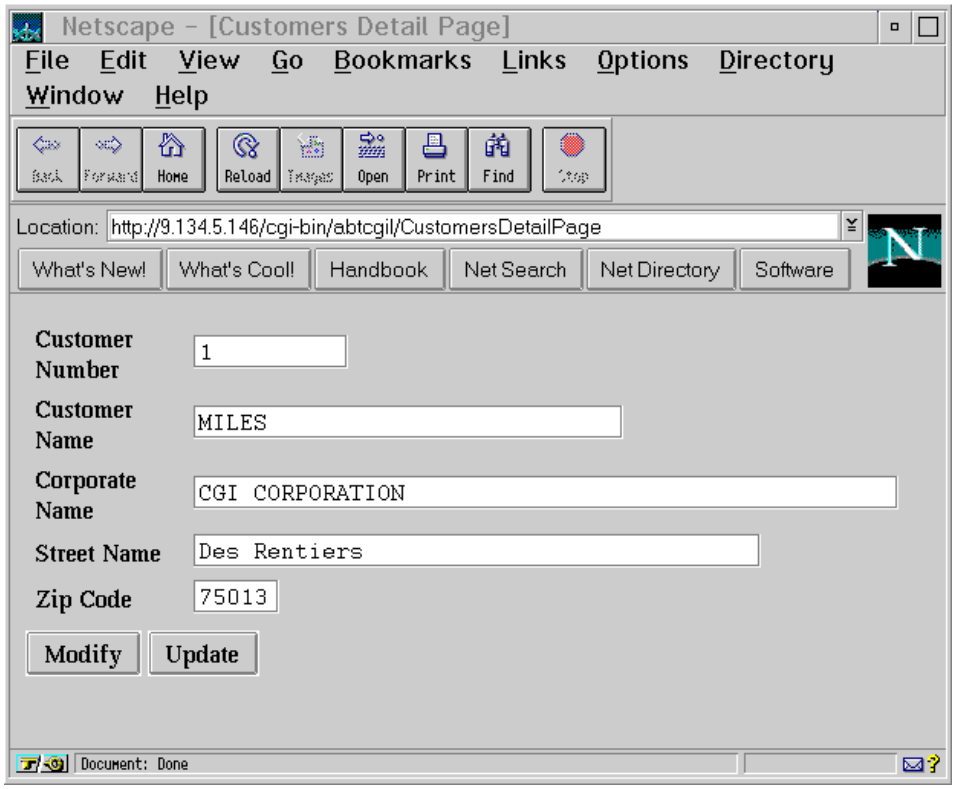
- **La page Customers**



Les fonctionnalités de cette page sont les suivantes :

- affichage, dès l'ouverture de la page, de la liste des clients dans un tableau de plusieurs colonnes, chaque colonne correspondant à une Rubrique de la Vue Logique. Le tableau étant un composant statique, l'utilisateur n'a pas la possibilité de sélectionner un client dans cette liste.
- affichage d'un attribut de la Proxy Elémentaire, en l'occurrence le numéro de client, dans une liste qui permet la sélection.
- navigation vers une seconde page qui présente la fiche du client sélectionné lorsque l'utilisateur clique sur le bouton **`Show Detail...`**.

- **La page Customers Detail**



Les fonctionnalités de cette page sont les suivantes :

- affichage des informations correspondant au client précédemment sélectionné dans la page **`Customers`** dans des champs de mise à jour.
- modification de ces informations : l'utilisateur doit cliquer sur le bouton **`Modify`** une fois les modifications saisies dans les différents champs.
- mise à jour de la base de données par la prise en compte des modifications via le bouton **`Update`**.

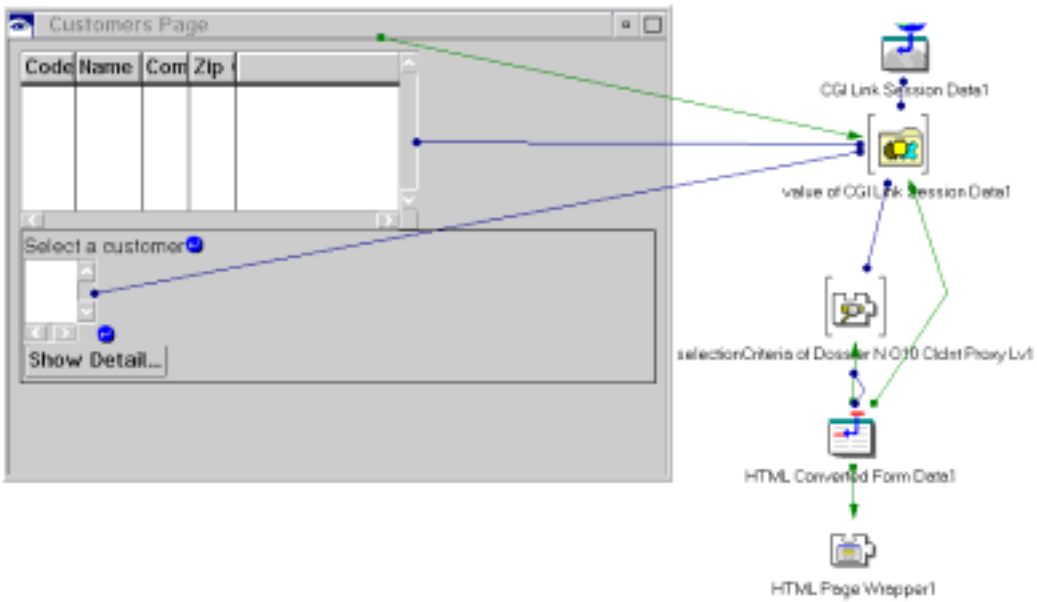
6.3.4. Exemple : Développement de l'interface utilisateur

Cette section ne détaille ni l'utilisation du Composition Editor ni l'utilisation standard du Web Connection et des parts HTML. Pour plus de détails, reportez-vous aux différents manuels associés à VisualAge mentionnés au début du manuel.

La réalisation de cet exemple présuppose en particulier la lecture du manuel *VisualAge Web Connection User's Guide*.

6.3.4.1. Développement de la page Customers

Une fois la programmation de cette page terminée, le Composition Editor se présente comme ci-dessous :



Pour parvenir à ce résultat, les opérations présentées ci-après sont nécessaires.

- **Gestion du contexte et intégration de la Proxy Vue de Dossier dans le Composition Editor**

Après l'initialisation de la Page HTML, vous devez intégrer la PVD dans le Composition Editor.

L'existence de deux pages dans l'application rend nécessaire l'utilisation d'un part **`CGI Link Session Data`** dans chacune des Pages afin de gérer le contexte applicatif au cours d'un même dialogue avec un utilisateur.

- Dans la palette des parts, sélectionnez la catégorie Web Connection, puis le part **`CGI Link Session Data`**. Placez ce part sur la Free Form Surface.
- Dans la fenêtre **`Settings`** de ce part, indiquez le nom de la Proxy Racine dans la zone **`Data type of the value`**.
- Faites un Tear-Off de l'attribut **`value`** du part **`CGI Link Session Data`** pour obtenir la Proxy Racine. Placez le part variable ainsi obtenu sur la Free Form Surface.

- **Programmation de l'affichage de la liste des clients dans une table HTML statique**

Cette table permet l'affichage des données dans un tableau semblable au container, mais contrairement à ce dernier, elle n'est pas interactive : l'utilisateur ne peut donc y sélectionner un client par exemple.

- Dans la palette des parts, catégorie Web Connection, sélectionnez le part **`Html Table`**. Placez- la dans la Page HTML.
- Puis sélectionnez le part **`HTML Table Column`** et placez-le dans le part **`HTML Table`**. Ouvrez la fenêtre **`Settings`** du part **`HTML Table Column`** :
  - dans la zone **`Heading`**, indiquez le libellé que vous souhaitez voir afficher pour la colonne dans l'application finale.
  - dans la zone **`Data type`**, indiquez le type de donnée de la Rubrique (Integer, String, Date...).
  - dans la zone **`Attribute to display`**, indiquez le code de la Rubrique correspondante.
- Créez de cette manière autant de parts **`HTML Table Column`** que vous souhaitez de colonnes, c'est-à-dire de Rubriques, dans votre table.
- Connectez la Proxy Racine, via son attribut **`rows`** au part **`HTML Table`**, via son attribut **`items`**.
- Pour que la table soit alimentée dès l'ouverture de la page, connectez la Proxy Racine, via l'action **`selectInstances`**, au part **`HTML Table`**, via l'événement **`aboutToGenerateHtml`**.

- **Utilisation d'un formulaire dans la page**

Les autres fonctionnalités de la page, à savoir l'affichage des Rubriques de type identifiant dans une liste qui permet la sélection et la navigation vers la page **`CustomersDetail`**, nécessite l'utilisation d'un formulaire ou part **`HTML Form`**.

Rappel Les formulaires rapprochent le comportement d'une application Web de celui d'une application GUI standard en permettant l'intégration de contrôles graphiques similaires dans une page HTML. Les formulaires permettent donc la réalisation d'applications Web dynamiques.

Pour utiliser un formulaire, effectuez les opérations suivantes :

- Insérez un part **`HTML Form`** dans la Page HTML.
- Dans la fenêtre **`Settings`** de ce part, vous devez :
  - dans la zone **`Server location`**, sélectionner le paramètre **`Part name`**.
  - dans la zone **`Name`**, entrer le nom de la Page HTML (dans notre exemple, il s'agit de **`CustomersPage`**). Vous spécifiez ainsi dans quelle page le part **`HTML Form`** est utilisé et par conséquent que les traitements effectués à partir des parts contenus dans le part **`HTML Form`** sont effectués dans la page indiquée.

- **Programmation de l'affichage des identifiants dans une liste dynamique**
  - Le libellé **Select a customer** est un part **HTML Text**.
  - Pour obtenir la liste effectuez un Quick HTML sur l'attribut **rows** de la Proxy Racine. Placez le part dans le part **HTML Form**. Dans la fenêtre **Settings** du part, les éléments suivants doivent être indiqués :
    - Dans la zone **Data type**, sélectionnez **Integer**.
    - Dans la zone **Attribute to display**, entrez le code de la Rubrique identifiant.

    Par défaut, le nom du part est **rowsList**.
  - La programmation de l'affichage des Rubriques de type identifiant dans cette liste comprend les étapes suivantes :
    - Connectez la Proxy Racine, via son attribut **rows**, à la liste, via l'attribut **items**.
    - Réalisez un Tear-Off de l'attribut **selectionCriteria** de la Proxy Racine et placez le part obtenu sur la Free Form Surface.

- **Programmation de l'alimentation de la page CustomersDetail**

Cette étape consiste à programmer l'alimentation de la page **CustomersDetail** avec les informations relatives au client sélectionné dans la liste lorsque l'utilisateur clique sur le bouton **Show Detail...**.

  - Insérez sur la Free Form Surface un part **HTML Converted Form Data**. Dans la fenêtre **Settings** de ce part, indiquez dans la zone **Name of the part containing the page** le nom de la page dans laquelle les traitements seront effectués (dans notre exemple, il s'agit de **CustomersDetailPage**).
  - Insérez un bouton, changez son libellé (ici **Show Detail...**), puis dans le menu contextuel de ce part, sélectionnez le choix **Change name...**. Dans la fenêtre qui s'ouvre, entrez un nom pour ce bouton (par exemple, **ShowDetailButton**).

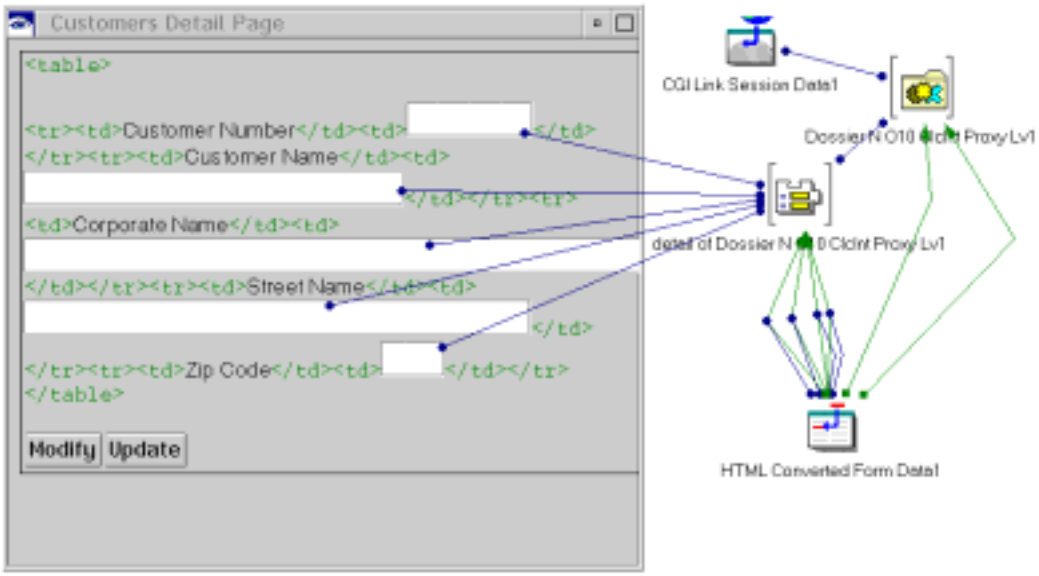
    L'interface publique du part **HTML Converted Form Data** se trouve alors enrichi de l'événement **ShowDetailButton clicked**.
  - Connectez le part **HTML Converted Form Data**, via l'événement **ShowDetailButton clicked**, au part **selectionCriteria** résultant du Tear-Off, via l'attribut correspondant au code de la Rubrique identifiant. Le lien nécessite un paramètre : connectez le part **HTML Converted Form Data**, via l'attribut **rowsList**, au lien via le paramètre **value**.
  - Connectez le part **HTML Converted Form Data**, via l'événement **ShowDetailButton clicked**, à la Proxy Racine, via l'action **readInstance**.

- **Programmation de la navigation vers la page CustomersDetail**
  - Insérez un part **HTML Page Wrapper**.
  - Connectez le part **HTML Converted Form Data**, via l'événement **ShowDetailButton clicked**, au part **HTML Page Wrapper**, via l'événement **transferRequest**.
  - Cette étape suppose que la page secondaire **CustomersDetail** est construite. Lors de la construction de cette page, l'attribut **self** de la Proxy Racine a été promu et le code de cet attribut promu est **dossierNO10ClclntProxyLv**.

De retour dans la page principale, connectez la Proxy Racine, via l'attribut **self**, au part **HTML Page Wrapper**, via l'attribut **dossierNO10ClcIntProxyLv**.

6.3.4.2. Programmation de la page CustomersDetail

Une fois la programmation de cette page terminée, le Composition Editor se présente comme ci-dessous :



- **Programmation de la gestion du contexte et intégration de la Proxy Vue de Dossier dans le Composition Editor**

Les opérations à effectuer dans cette étape étant identiques à celle décrites précédemment pour la page **Customers**, veuillez vous reporter à la section correspondante.

- **Programmation de l'affichage du détail d'un client**

Pour alimenter les champs de mise à jour avec les informations correspondant au client précédemment sélectionné :

- réalisez un Tear-Off de l'attribut **detail** de la Proxy Racine et placez le part obtenu sur la Free Form Surface. Par défaut, le nom de chaque champ de mise à jour généré est ainsi codifié : **CodeRubriqueField**, le suffixe **Field** étant invariable.
- ouvrez le menu contextuel de ce part et sélectionnez le choix Quick HTML. Dans la liste qui s'affiche alors, sélectionnez **self as HTML Table**.
- Placez le résultat dans le part **HTML Form**.

- **Programmation de la modification d'un client**

Pour cette étape et la suivante, vous devez utiliser un part **HTML Converted Form Data**. Reportez-vous à l'étape correspondante dans la section consacré au développement de la page **Customers**.

C'est un service de mise à jour locale assuré par le bouton **Modify**. Pour programmer ce bouton:

- Insérez un bouton, changez son libellé, puis changez son nom en **ModifyButton** par exemple.
- Vous devez ensuite :
  - connecter le part **HTML Converted Form Data**, via l'événement **ModifyButton clicked**, au part résultant du Tear-Off de l'attribut **detail** de la Proxy Racine, via l'attribut correspondant au code de la Rubrique associé à un champ de mise à jour.
  - puis, le lien créé nécessitant un paramètre, connecter le part **HTML Converted Form Data**, via l'attribut de nom **CodeRubriqueField** corespondant, au lien, via l'attribut **value**.

  Renouvelez ces deux opérations successives pour chaque champ de mise à jour.
- Connectez enfin le part **HTML Converted Form Data**, via l'événement **ModifyButton clicked**, à la Proxy Racine, via l'action **modifyInstance**.

- **Programmation de la mise à jour serveur**

La mise à jour serveur est assurée par le bouton Update. Pour sa programmation, vous devez :

- insérer un bouton, changer son libellé, puis changer son nom en **UpdateButton** par exemple.
- connecter le part **HTML Converted Form Data**, via l'événement **UpdateButton clicked**, à la Proxy Racine, via l'action **updateFolder**.

6.4. Gestion des erreurs

Il n'y a pas de message d'erreur lors des phases d'extraction et de génération.

Il est donc indispensable d'avoir compilé correctement le Composant Applicatif que l'on veut extraire car

- la commande GVC de la procédure GPRT extrait toujours sans afficher d'erreur, même si le Composant Applicatif ne contient aucune Vue Logique,
- le générateur ne fait aucun contrôle sur le fichier qu'il prend en entrée.

  En revanche, des messages d'erreur peuvent apparaître lors des phases de développement, de test et d'exécution de l'application. Ces messages proviennent d'erreurs locales, serveur ou de communication.

6.4.1. Principes

6.4.1.1. Introduction

Dans VisualAge, la gestion des erreurs est assurée par le gestionnaire d'erreurs à travers une classe générique qui manipule tous les types d'erreur.

Il existe quatre types d'erreur :

- les erreurs locales, émises par le composant Client, qui correspondent à des erreurs de manipulation ou de saisie dans le Composant Client,
- les erreurs serveur, envoyées par le composant Applicatif, qui sont des erreurs d'accès aux données et des erreurs utilisateur positionnées dans le serveur,
- les erreurs système, envoyées aussi par le Composant Applicatif, qui correspondent à un déphasage entre la Proxy et les Composants Applicatifs.
- les erreurs de communication.

L'interface publique du gestionnaire d'erreur contient tous les attributs et les événements nécessaires à la programmation des erreurs dans l'application graphique.

Vous pouvez personnaliser tous les libellés d'erreur dans un fichier dédié.

6.4.1.2. Gestion événementielle

6.4.1.2.1. Services de mises à jour locales

Chaque service de mise à jour locale émet après son exécution un événement caractérisant le résultat du traitement exécuté :

- `noError` lorsque le traitement s'est correctement exécuté.
- `localError` lorsque le traitement ne s'est pas correctement exécuté. Dans ce cas la liste des messages d'erreur et la liste de leur clé sont transférées dans le gestionnaire d'erreurs.

6.4.1.2.2. Services de mise à jour serveur

Chaque service de mise à jour serveur émet après son exécution un événement caractérisant le résultat de son exécution. Les différents événements susceptibles d'être émis sont les suivants :

- **noError** lorsque le traitement s'est correctement exécuté.
- **serverError** caractérisant une erreur d'accès à la base ou erreur utilisateur.
- **systemError** caractérisant une erreur de synchronisation entre les descriptions contenues dans les différents composants de l'application.
- **fatalError** caractérisant une erreur de communication.

6.4.1.3. Programmation visuelle

L'accès aux erreurs est assuré par l'attribut **errorManager** de la Proxy Racine. Vous devez donc faire un Tear-Off sur cet attribut.

L'attribut **errorManager** contient une instance de la classe **VapErrorManager** qui manipule des instances de la classe **VapError**. Une instance de la classe **VapError** représente une erreur locale, serveur, système ou de communication.

L'instance de la classe **VapErrorManager** contient, dans son attribut **label**, le libellé associé à l'erreur. Ce libellé est lu dans le fichier des erreurs **vaperror.txt** installé dans le répertoire de VisualAge. Ce fichier contient tous les libellés d'erreur, que l'erreur soit locale, serveur, système ou de communication. Vous pouvez personnaliser les libellés d'erreur à partir de ce fichier.

☞ Les erreurs serveur sont, en principe, retournées par le serveur. Mais lorsque l'erreur est telle que le serveur ne peut la retourner ou lorsque le libellé est à blanc dans le fichier des libellés d'erreur du Référentiel, le gestionnaire d'erreurs va chercher le libellé correspondant à la clé retournée dans le fichier **vaperror.txt**.

Pour des informations sur la personnalisation des erreurs côté serveur, reportez-vous au Guide Utilisateur Pacbench C/S, *Vol. II - Services Applicatifs*.

La gestion applicative des erreurs renvoyées par le gestionnaire d'erreurs est basée sur la création de connexions entre les différents événements associés à l'absence ou la présence d'erreurs et des actions du part graphique.

Il est indispensable de connecter au minimum les événements **localError**, **serverError**, **systemError** et **fatalError** sur l'action **openWidget** d'une fenêtre qui affiche l'attribut **errorList**.

6.4.1.4. Possibilité de personnaliser les libellés standard

Tous les libellés d'erreur sont contenus dans le fichier **vaperror.txt**, installé dans le même répertoire que VisualAge. A l'initialisation du système, ce fichier permet de générer un dictionnaire d'instances d'erreurs gérées par la classe **VapErrorManager**. Le développeur a la possibilité de personnaliser ou d'ajouter ses propres libellés d'erreurs. Ces libellés peuvent être :

- des erreurs détectées dans le traitement client,
- des erreurs utilisateur détectées dans le serveur (opérateur **ERU**).

En cas de réinstallation, veillez à sauvegarder le fichier **vaperror.txt** que vous avez personnalisé. Les éventuelles nouveautés contenues dans le fichier chargé à l'installation sont indiquées dans le fichier **readme**. Vous devez copier ces modifications et les coller dans votre fichier personnalisé. Remplacez ensuite le nouveau fichier par votre propre fichier.

Chaque erreur est décrite, dans le fichier **vaperror.txt**, sur cinq lignes. Le libellé français est présent sur la ligne commençant par **FLAB** et le libellé anglais est présent sur la ligne commençant par **ELAB**. Pour personnaliser un message d'erreur, il vous suffit d'écraser la ligne correspondante.

Voici un exemple de description d'une erreur telle que présentée dans ce fichier :

```
ERR:UnknownInstance
FLAB:Instance inconnue
ELAB: Unknown instance
KEY:,,,'UNKNOW',,'L'
TYP:LocalError
```

Vous pouvez donc, par exemple, remplacer "Instance inconnue" par "sélection d'une instance inconnue", ou bien encore remplacer le libellé anglais par un libellé dans une autre langue.

6.4.1.5. Structure de la clé d'erreur

Vous devez connaître la structure de la clé d'erreur si vous voulez gérer vos libellés dans un fichier autre que le fichier **VAPERROR.TXT** fourni. Il vous incombera alors de mettre en correspondance un libellé et la clé spécifique à chaque erreur.

Les erreurs décrites dans ce tableau sont celles qui sont les plus susceptibles d'apparaître.

Col 1-3	Col 4-9	Col 10-13	Col 14-19	Col 20	Col 21	Col 22-25	
bib	ser	seg			S	DUPL	Création à tort [S]
bib	ser	seg			S	NFND	Modif/annul à tort [S]
bib	ser				S	code erreur	Erreur utilisateur [S]
bib	ser	vue	rub	2	S		Rubrique obligatoire [S]
bib	ser	vue	rub	5	S		Erreur de valeur [S]
bib	ser	vue	ACCESS		S		Erreur d'accès données [S]
bib	ser	STRU			S		Erreur structure vue [S]
bib	ser	VERS			S		Erreur version [S]
bib	ser	VIEW			S		Vue inconnue [S]
bib	ser	SERV			S		Service inconnu [S]
bib	ser	LTH			S		Longueur vue erronée [S]
			MISPCV		S		Déphasage composants [S]
		OPEN			C		Erreur open serveur [C]
		CALL			C		Erreur appel serveur [C]
		CLOS			C		Erreur close serveur [C]
			CRDVIO		L		Cardinalités non respectées [L]
			CREATE		L		Création invalide [L]
			DELETE		L		Suppression invalide [L]
			EXTR		L		Méthode d'extraction invalide [L]
			FOLDER		L		Dossier en lecture seule [L]
			HIERAR		L		Violation hiérarchie [L]
			INVINS		L		Instance erronée [L]
			LENGTH		L		Erreur de longueur [L]
			LOCATE		L		Environnement serveur indisponible [L]
			LOCKED		L		Déjà verrouillée [L,S]
			MISINS		L		Instance absente [L]
			MISPAR		L		Instance père absente [L]
			MISREF		L		Instance refer absente [L]
			MISUSR		L		Instance user absente [L]
			MODIFY		L		Modification invalide [L]
			NLOCIN		L		Instance ne peut être verrouillée [L]
			NTLOCK		L		Déverrouillage à tort [L,S]
			REQUIR		L		Rubrique obligatoire [L]
			SUBSCH		L		Rubrique n'appartient pas au sous-schéma [L]
			SUSER		L		Service utilisateur invalide [L]
			UNKNOW		L		Instance inconnue [L]
			UPDTRQ		L		Mise à jour serveur nécessaire [L]
			VALUE		L		Erreur de valeur [L]

Légende bib = code bibliothèque vue = code Vue Logique ser = code serveur

rub = code Rubrique seg = code segment d'accès physique

[S] = erreur serveur [L] = erreur locale [C] = erreur de communication

6.4.2. Liste des erreurs

6.4.2.1. Erreurs locales

Les messages d'erreur locaux sont listés ci dessous :

- **`Instance inconnue`**

Ce message apparaît en cas de sélection d'une instance inconnue du cache local.

- **`Violation hiérarchie`**

Ce message apparaît en cas de sélection d'une instance de noeud dépendant alors qu'elle ne dépend pas de l'instance contenue dans l'attribut **`detail`** du noeud père.

- **`Instance père absente`**

Ce message apparaît en cas de sélection d'une instance de noeud dépendant alors que l'instance supérieure est inexistante.

- **`Instance déjà verrouillée`**

Ce message apparaît en cas de demande de verrouillage d'une instance déjà verrouillée.

- **`Instance non verrouillée`**

Ce message apparaît en cas de demande de déverrouillage sur une instance déjà déverrouillée.

- **`Instance absente`**

Ce message apparaît quand une action est appliquée à l'attribut **`detail`** alors que cet attribut ne contient pas d'instance.

- **`Instance user absente`**

Ce message apparaît quand une action utilisateur est appliquée à l'attribut **`userDetail`** alors que cet attribut ne contient pas d'instance.

- **`Instance référençante absente`**

Ce message apparaît quand l'action **`transferReferenceFrom-SelectedRow:`** est appliquée à une instance de noeud référence alors qu'aucune instance n'est sélectionnée dans l'attribut **`detail`** du noeud référençant.

- **`Instance erronée`**

Ce message apparaît en cas de contrôle des valeurs non respecté.

- **`Dossier en lecture seule`**

Ce message d'erreur apparaît quand une action de mise à jour est appliquée à un Dossier non verrouillé alors que le verrouillage logique est positionné côté Serveur. Pour pouvoir mettre à jour, l'utilisateur doit au préalable verrouiller l'instance de Dossier.

- **Création invalide**

Ce message apparaît en cas de création d'une instance qui existe déjà dans le cache local.

- **Modification invalide**

Ce message apparaît lorsque l'instance à modifier n'existe pas dans le cache local.

- **Suppression invalide**

Ce message apparaît lorsque l'instance à supprimer n'existe pas dans le cache local.

- **Mise à jour serveur nécessaire**

Ce message apparaît lorsqu'une action est appliquée à une instance alors que l'instance supérieure qui a été créée localement n'existe pas encore dans la base. Une mise à jour serveur préalable de l'instance supérieure est requise.

- **Cardinalités non respectées**

Ce message d'erreur apparaît en cas de non respect des cardinalités lors de l'activation d'une action de mise à jour.

- **L'instance ne peut être verrouillée**

Ce message apparaît lorsque l'action de verrouillage est utilisée sur une instance alors que le verrouillage n'est pas implémenté dans le serveur.

- **Rubrique obligatoire**

Lors de la création ou modification d'une instance, une des Rubriques de la Vue Logique autre qu'une Rubrique clé n'est pas renseignée.

- **Rubrique n'appartient pas au sous-schéma**

Ce message apparaît lorsqu'une Rubrique de l'instance courante est modifiée alors qu'elle n'a pas été renseignée pour cette instance à l'issue d'un accès serveur paramétré avec un sous-schéma.

- **Erreur de valeur**

Ce message apparaît lorsque la valeur renseignée pour une Rubrique ne figure pas dans la table de valeurs de cette Rubrique. Cette erreur ne peut apparaître si l'utilisateur sélectionne une des valeurs proposées dans la table.

- **Erreur de longueur**

Cette erreur se produit quand le contenu d’une rubrique de l’instance courante a une taille supérieure à la taille autorisée pour la rubrique.

- **Environnement serveur indisponible**

Ce message apparaît à l'initialisation du middleware, si les paramètres nécessaires à sa mise en oeuvre sont incorrects.

- **Contexte inconnu**

Ce message apparaît lorsque l’identifiant de réponse transmis n’est pas connu de la Proxy pour la localisation courante lors de l’exécution des actions **`getReplyOf:aReplyId`** et **`resetPendingReplyOf:aReplyId`**.

- **`Action incompatible avec le protocole utilisé`**

Ce message apparaît lors de la demande d’exécution d’une requête interdite.

- **`Nombre de réponses en attente dépassé`**

Ce message apparaît lorsque le compteur de réponses en attente pour une requête asynchrone est dépassé. La requête concernée n’est dans ce cas pas émise.

6.4.2.2. Erreurs Serveur

Ces messages d’erreur sont générés en standard par les Services Applicatifs. Ils se répartissent en trois types :

- Erreurs d'accès irrécupérables : **`Erreur d'accès données`**

Il s'agit des erreurs d'accès irrécupérables à un fichier ou une base de données relationnelle.

- Erreurs logiques d'accès : **`Création à tort`** et **`Modif/Annul à tort`**.

- Erreurs de contrôle des champs de la Vue Logique : **`Erreur de valeur`** et **`Rubrique obligatoire`**.

6.4.2.3. Erreurs Système

Les erreurs système sont les suivantes :

- **`Service inconnu`**

Ce message apparaît lorsque le service demandé par la Proxy n'est pas reconnu par le Composant Applicatif.

- **`Longueur de vue erronée`**

Ce message apparaît lorsqu'il y a un changement de format dans une Vue Logique associée à la Proxy et que celle-ci n'a pas été regénérée.

Pour résoudre ce problème, vous devez regénérer la Proxy.

- **`Déphasage composants`**

Cette erreur survient lorsqu'il y a un déphasage entre les composants Client et Serveur.

Pour résoudre ce problème, vous devez regénérer la Proxy.

6.4.2.4. Erreurs système générées par le Moniteur de Communication ou le Gestionnaire de Services

Ces erreurs sont, pour la plupart, des erreurs internes que vous résoudrez en contactant le support VisualAge Pacbase.

6.4.2.4.1. Erreurs générées par le Moniteur de Communication

Col 1-3	Col 4-9	Col 10-13	Col 21	
bib	mon	LSRV	S	Erreur de longueur du message reçu
bib	mon	PNUM	S	Erreur de structure du paramètre "numéro de service"
bib	mon	PCOD	S	Erreur de structure du paramètre "code de service"
bib	mon	PNOD	S	Code du Dossier inconnu du Moniteur de Communication
bib	mon	PCVS	S	Erreur de structure d'une demande de service en provenance du client
bib	mon	PCVF	S	Erreur de structure du message en provenance du client
bib	mon	WF00	S	Erreur d'accès au fichier de travail

Légende bib = code bibliothèque mon = code Moniteur de Communication S = erreur système

6.4.2.4.2. Erreurs générées par le Gestionnaire de Services

Col 1-3	Col 4-9	Col 10-13	Col 21	
bib	ges	SRV1	S	Service non trouvé dans le fichier de travail
bib	ges	LNG1	S	Erreur de conversion de la longueur du service sur une requête multi-messages
bib	ges	LNG2	S	Erreur de conversion de la longueur du service sur une requête mono-message
bib	ges	NOS1	S	Erreur de structure du paramètre "numéro de service"
bib	ges	NOS2	S	Erreur de conversion du paramètre "numéro de service"
bib	ges	SRV1	S	Erreur de structure du paramètre "code service"
bib	ges	SRV2	S	Code service inconnu dans le Dossier
bib	ges	DOS1	S	Paramètre "nom du Dossier" absent
bib	ges	DOS2	S	Erreur de longueur du paramètre "nom du Dossier"
bib	ges	VER2	S	Erreur de longueur du pramètre "numéro de version"
bib	ges	NOD1	S	Paramètre "nom du noeud" absent
bib	ges	NOD2	S	Erreur de longueur du paramètre "nom du noeud"
bib	ges	NOD3	S	Nom du noeud inconnu dans le Dossier
bib	ges	TYNO	S	Service non autorisé sur le noeud
bib	ges	SCH1	S	Erreur de structure du paramètre "code sous-schéma"
bib	ges	SCH2	S	Code sous-schéma inconnu dans le Dossier
bib	ges	NOCP	S	Erreur de structure du paramètre "nombre d'occurrences"
bib	ges	NOC1	S	Erreur de longueur sur le paramètre "nombre d'occurrences"
bib	ges	NOC2	S	Erreur de conversion du paramètre "nombre d'occurrences"
bib	ges	EXT1	S	Erreur de structure du paramètre "méthode d'extraction"
bib	ges	EXT2	S	Méthode d'extraction inconnue dans le Dossier
bib	ges	USR1	S	Paramètre "service utilisateur" absent
bib	ges	USR2	S	Erreur de longueur sur le paramètre"service utilisateur"
bib	ges	USR3	S	Service Utilisateur inconnu dans le Dossier
bib	ges	CHK1	S	Paramètre "Check Option" absent
bib	ges	CHK2	S	Erreur de longueur sur le paramètre "Check Option"
bib	ges	RFH1	S	Paramètre "Refresh Option" absent
bib	ges	RFH2	S	Erreur de longueur sur le paramètre"Refresh Option"
bib	ges	LCK1	S	Paramètre "Lock Timestamp" absent
bib	ges	LCK2	S	Erreur de longueur sur le paramètre "Lock Timestamp"

Col 1-3	Col 4-9	Col 10-13	Col 21	
bib	ges	PCV1	S	Erreur de structure du paramètre "Selection Criteria"
bib	ges	PC01	S	Erreur de conversion du paramètre "Selection Criteria"
bib	ges	PILO	S	Erreur d'accès à l'enregistrement pilote du fichier de travail
bib	ges	BUF1	S	Erreur de structure du buffer utilisateur
bib	ges	PC02	S	Erreur de conversion d'un champ du buffer utilisateur
bib	ges	FRWR	S	Erreur d'accès en écriture au fichier de travail
bib	ges	FRW2	S	Erreur d'écriture du dernier enregistrement du fichier de travail
bib	ges	FRRE	S	Erreur d'accès en lecture au fichier de travail avant mise à jour
bib	ges	FRRD	S	Erreur d'accès en lecture au fichier de travail
bib	ges	FRRW	S	Erreur d'accès en mise à jour au fichier de travail
bib	ges	CP01	S	Erreur de structure d'un champ de "Selection Criteria" en provenance du Composant Applicatif
bib	ges	CP02	S	Erreur de structure d'un champ d'une instance de Vue Logique en provenance du Composant Applicatif
bib	ges	ERKY	S	Erreur de structure de la clé du "Selt Message" en provenance du Composant Applicatif
bib	ges	ERLA	S	Erreur de structure du label du "Selt Message" en provenance du Composant Applicatif
bib	ges	PCV!	S	Erreur de structure d'un champ du buffer utilisateur en provenance du Composant Applicatif
bib	ges	PCV3	S	Erreur de structure d'un champ d'une instance de Vue Logique en provenance du client
bib	ges	PC03	S	Erreur de conversion d'un champ d'une instance de Vue Logique en provenance du client
bib	ges	PCV4	S	Erreur de structure d'un champ de "Selection Criteria" en provenance du client
bib	ges	PC05	S	Erreur de conversion d'un champ de "Selection Criteria" en provenance du client
bib	ges	NUVE	S	Erreur du paramètre "numéro de version" dans le Composant Applicatif élémentaire
bib	ges	STRU	S	Erreur du paramètre "structure" dans le Composant Applicatif élémentaire
bib	ges	VIEW	S	Erreur du paramètre "code de la Vue Logique" dans le Composant Applicatif élémentaire
bib	ges	SERV	S	Erreur du paramètre "code opération" dans le Composant Applicatif élémentaire
bib	ges	LTH	S	Erreur du paramètre "longueur" dans le Composant Applicatif élémentaire
bib	ges	PCV5	S	Erreur de structure du code action d'une instance de Vue Logique à mettre à jour
bib	ges	PCV6	S	Erreur de structure d'un champ d'une instance de Vue Logique à mettre à jour en provenance du client
bib	ges	PCV7	S	Erreur de structure du vecteur de présence d'un champ d'une instance de Vue Logique à mettre à jour
bib	ges	PC06	S	Erreur de conversion d'un champ d'une instance de Vue Logique à mettre à jour en provenance d'un Composant Applicatif élémentaire
bib	ges	ERK1	S	Erreur de structure de la clé de message d'erreur utilisateur
bib	ges	ERL1	S	Erreur de structure du label de message d'erreur utilisateur
bib	ges	OCNB	S	Erreur de structure, dans le message d'erreur utilisateur, du code champ sur lequel porte l'erreur
bib	ges	DANA	S	Erreur de structure du code de champ erroné dans le message d'erreur utilisateur
bib	ges	PCVS	S	Erreur de structure d'une demande de service en provenance du client
bib	ges	PCVF	S	Erreur de structure du message en provenance du client
bib	ges	WF00	S	Erreur d'accès au fichier de travail

Légende bib = code bibliothèque ges = code Gestionnaire de services S = erreur système

6.4.2.5. Erreurs de Communication

Si un message d'erreur de communication apparaît, vous devez d'abord en informer le responsable de la communication car la cause peut être un encombrement de ligne, une ligne défectueuse, un serveur indisponible...

Trois messages d'erreur de communication sont susceptibles de s'afficher :

- `Erreur open serveur`
- `Erreur appel serveur`
- `Erreur close serveur`

6.4.3. Exemple

Pour illustrer la gestion des erreurs dans une application cliente, nous allons nous baser sur l'application que nous avons utilisée dans ce chapitre, sous-chapitre **6.2** *Exemple d'une application graphique standard*. Supposons que vous souhaitez programmer l'ouverture d'une fenêtre de messages d'erreurs intitulée `Error Manager` lorsque le Gestionnaire d'erreurs détecte une erreur. Cette fenêtre affiche la liste et le nombre de messages d'erreurs renvoyés lors d'une action de mise à jour serveur. Elle permet également de restaurer le contexte de chaque erreur.

- **Programmation de l'affichage de la liste et du nombre de messages d'erreur**

Cette fenêtre est développée dans la même Vue VisualAge que la fenêtre `Orders`. La programmation comprend les étapes suivantes :

  - Pour accéder à l'interface publique du Gestionnaire d'erreurs, vous devez réaliser un Tear-Off de l'attribut `errorManager` de la Proxy Racine. Posez le part correspondant sur la Free Form Surface.
  - Pour afficher le nombre d'erreurs, réalisez un Tear-Off de l'attribut `errorCount` du Gestionnaire d'erreurs.
  - Pour afficher la liste des messages d'erreurs, effectuez un Quick-Form de l'attribut `errorList` du Gestionnaire d'erreurs et insérez le container obtenu dans la fenêtre `Error Manager`. Dans notre exemple, nous ne conservons que l'attribut `label` de `errorList`.
  - Pour demander au Gestionnaire d'erreurs de retourner les libellés d'erreurs dans la liste, vous devez le connecter, via les événements `localError`,`serverError`, `systemError`, `fatalError` à la fenêtre `Error Manager`, via l'action `openWidget`.

- **Programmation de la restauration du contexte d'erreur**

La restauration du contexte d'erreur s'applique à une instance à la fois.

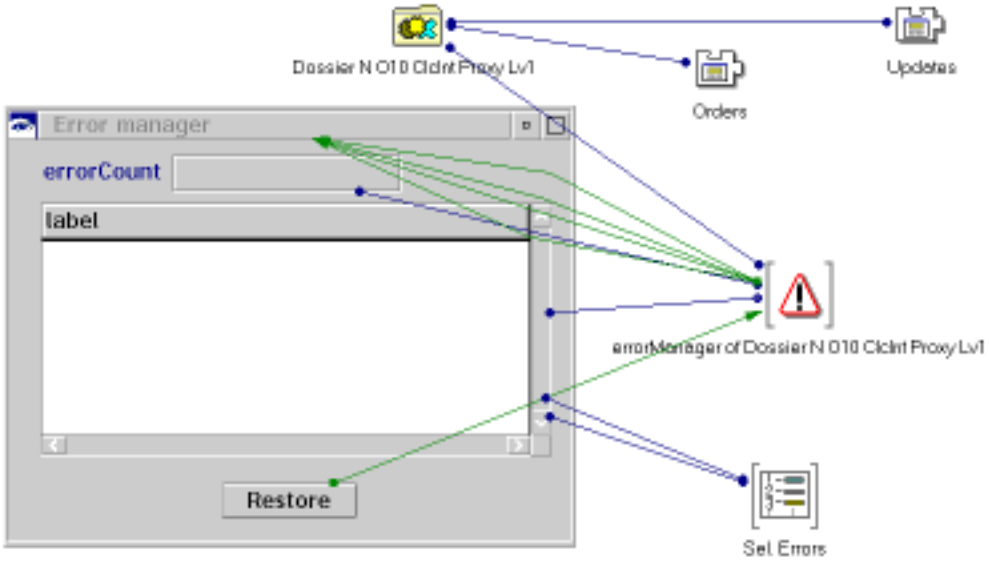
Il s'agit tout d'abord de transférer dans un Ordered Collection les items sélectionnés du tableau, chaque item correspondant à une instance de noeud à l'origine de l'erreur, et d'alimenter le détail de chaque noeud avec le premier item de cet Ordered Collection.

Vous devez pour cela :

- effectuer un Tear-Off de l'attribut **`selectedItems`** de la liste et placer l'Ordered Collection sur la Free Form Surface.
- Vous devez ensuite connecter le bouton **`Restore`**, via l'événement **`clicked`**, au Gestionnaire d'erreurs, via l'action **`restoreContextOfSelectedError:`**. Le lien apparaît en pointillé car l'action attend un paramètre. Pour passer le paramètre nécessaire, connectez l'Ordered Collection, via l'attribut **`first`** au lien, via l'attribut **`anError`**.

- **Résultat dans le Composition Editor**

La programmation de la fenêtre **`Error Manager`** terminée, le Composition Editor doit se présenter comme ci-dessous :



6.5. Gestion de la communication

6.5.1. Composants du Middleware

Le middleware de Pacbench Client/Serveur correspond à une application C et C++ dont les fonctions sont réparties dans trois types de DLLs :

- le premier type regroupe les fonctions d'interfaçage avec les langages C et C++ et les fonctions d'interprétation des services du middleware et d'aiguillage vers des librairies de fonctions spécialisées pour un protocole de communication spécifique.
- le deuxième type regroupe les fonctions d'exécution des services du middleware de Pacbench Client/Serveur pour un protocole de communication spécifique.
- le troisième type regroupe des fonctions de gestion de ressources telles que la traduction en clair des exceptions rencontrées.

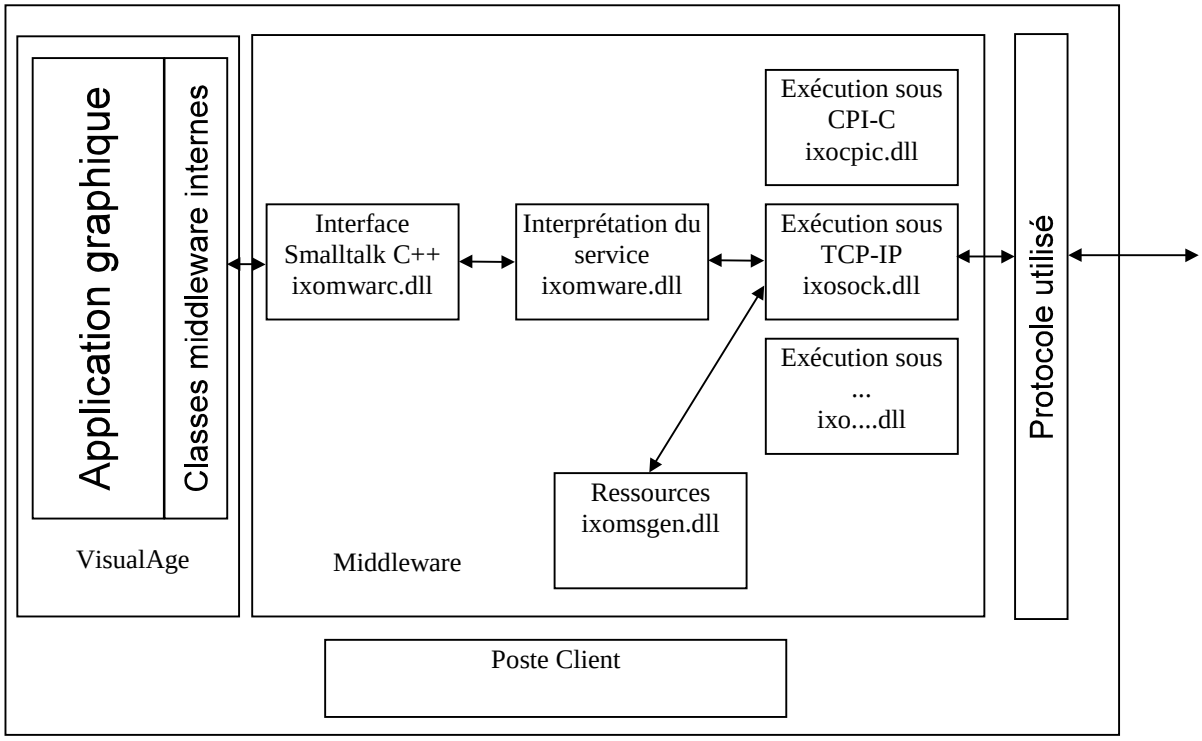
La version OS/2 du middleware de Pacbench Client/Serveur utilise également un run-time matérialisé par une DLL de code **CSOOM30**.

Nom de la DLL	Fonction	Plate-forme	Protocole
ixomwarp.dll	Type-1	Windows 95/NT OS2	Tous
ixomware.dll	Type-1	Windows 95/NT OS2, UNIX	Tous
ixocics.dll	Type-2	Windows 95/NT OS2	CICS-ECI
ixocpic.dll	Type-2	Windows 95/NT OS2	CPI-C
ixotux.dll	Type-2	Windows 95/NT UNIX	TUXEDO
ixosock.dll	Type-2	Windows 95/NT OS2, UNIX	TCP-IP Socket VA Pacbase
ixoloc.dll	Type-2	Windows 95/NT OS2	Appel de DLL COBOL
ixomqs.dll	Type-2	Windows 95/NT OS2, UNIX	MQSERIES
ixotmvs.dll	Type 2	Windows 95/NT, OS2, UNIX	TCP-IP Socket CICS
ixotcis.dll	Type 2	Windows 95/NT, OS2, UNIX	TCP-IP Socket TCIS
ixomsgen.dll	Type-3	Windows 95/NT OS2	Tous
csoom30.dll	runtime C++	OS2	Tous

Windows(*) = toutes plate-formes Windows

6.5.2. Schéma de traitement d'une requête

Dans VisualAge, les services du middleware sont exécutés à partir d'un ensemble de classes de communication spécifiques livrées à l'installation du produit.



6.5.3. Définition du contexte d’utilisation

La gestion de la communication nécessite la définition du contexte d'utilisation du middleware. Ce contexte est défini dans un fichier spécifique dans lequel sont indiqués, pour chaque Dossier généré, une ou plusieurs localisations.

Une localisation, ou *location* permet de spécifier les éléments qui autorisent les échanges de données entre les applications clientes et les gestionnaires de services et ce, que ce soit dans une étape de développement de l’application cliente ou lors de son exploitation. Par exemple, une application cliente peut s’adresser à un serveur sans trace pour une version d’exploitation.

Pour un Dossier, une localisation est composée :

- de l’identifiant de la localisation. Cet identifiant est sensible à la distinction majuscules/minuscules.
- du nom externe du moniteur de communication.
- de la longueur physique maximum d’un message d’échange.
- d’un nombre variable de paramètres qui permettent de faire fonctionner le middleware (adresse IP, trace...).

Les localisations sont spécifiées dans le fichier **`VAPLOCAT.INI`** (ce nom de fichier en majuscules est réservé).

☞ Le fichier utilisé par défaut est celui se trouvant dans le répertoire de démarrage lors du lancement de l’image VisualAge Smalltalk.

6.5.3.1. Structure du fichier VAPLOCAT.INI

Le fichier des localisations est structuré en sections et sous-sections.

Une section correspond à un Dossier et est identifiée par le code de celui-ci, placé entre [...].

Une sous-section correspond à une localisation et est identifiée par le nom de l'environnement spécifié pour l'option **`LOCATION`** dans la fenêtre **`Commentaires`** du Dossier. L'identifiant d'une sous-section est placé entre <...>.

Dans une sous-section, les différents paramètres sont représentés par des mots-clés.

Les mots-clés utilisés sont les suivants :

Mot-clé	Signification
COMMENT	Commentaire
LENGTH	Longueur physique du message (générée)
MONITOR	Nom externe du moniteur de communication (généré)
MWADDRESS	Adresse du serveur (pour SOCKET, TCPMVS, TCIS, TCPIMSI et TCPIMSE) Obligatoire pour ces protocoles Les formats suivants sont possibles: - Codification décimale : * adresse IP ou alias de l'adresse IP * blanc * numéro de port en décimal - Codification hexadécimale: * [1, 2]: '0x'' * [3, 6]: domaine AF_INET * [7,10]: numéro de port en hexadécimal * [11, 18]: adresse IP
MWARE	Protocole de communication **Valeurs** : **Protocoles** TUXEDO TUXEDO XA ou non XA CICS CICS ECI extend ou nonextend LOCAL Serveur sur la station Client CPIC CICS CPI-C/APPC ou IMS CPI-C/APPC SOCKET TCP-IP socket pour UNIX ou WINDOWS NT TCIS TCIS MQSERIES MQSERIES TCPMVS TCP-IP SOCKET pour MVS/CICS
MWBUFFERTYPE	Type de buffer (pour TUXEDO) Valeurs : - CARRAY (défaut) - FML
MWCODEPAGE	Code page du serveur - Facultatif
MWFMLNAME	Nom de fichier FML (pour TUXEDO). Attention : vous devez aussi indiquer le chemin dans la variable d'environnement FLDTBLDIR et le nom du fichier dans la variable d'environnement FIELDTBLS.
MWMAXREPLY	Nombre maximum de requêtes asynchrones en attente de réponse - Facultatif - Numérique
MWQUEUEMANAGER	Nom du manager de queues pour protocole asynchrone (MQSERIES) - Obligatoire pour ce protocole - Alphanumérique - 48 caractères max.
MWREPLYQUEUE	Nom de la queue de message en réception pour protocole asynchrone (MQSERIES) - Obligatoire pour ce protocole - Alphanumérique - 48 caractères max.
MWREQUESTEXPIRY	Durée de vie d'une requête asynchrone (en secondes) – Facultatif (illimité par défault) - Numérique
MWREQUESTQUEUE	Nom de la queue de message en émission pour protocole asynchrone (MQSERIES) - Obligatoire pour ce protocole - Alphanumérique - 48 caractères max.
MWTIMEOUT	Time out géré par le middleware (en secondes) - Facultatif - Numérique - Valeur: [0, 32767]
MWTRANSID	Identification du code transaction CICS sur 4 caractères : - obligatoire pour le protocole CICS Sockets TCP/IP

Mot-clé	Signification
	- facultatif pour le protocole CICS/ECI • Si le paramètre MWTRANSID est absent ou non valorisé, la transaction miroir CPMI est utilisée par défaut et est exécutée sur le serveur CICS. • Si le paramètre est valorisé, le code transaction est exécuté sur le serveur CICS pour chaque requête utilisant cette location. Cette transaction doit être définie sur le serveur comme une transaction miroir CICS et doit être associée au programme DFHMIRS. • Dans tous les cas, le paramètre MONITOR est obligatoire et doit correspondre à un programme exécutable sur le serveur CICS.
IXO_TRANSID(*)	Identification d code transaction CICS sur 4 caractères : - facultatif et utilisé avec le protocole CICS/ECI • Si le paramètre IXO_TRANSID est absent ou non valorisé, le code transaction CPMI est utilisé par défaut et est exécuté sur le serveur CICS. • Si le paramètre est valorisé, le programme est appelé par la transaction miroir CPMI, mais est relié sous le nom du code transaction spécifié dans la valeur IXO_TRANSID. Peut être utilisé pour des problèmes de sécurité par exemple avec les plans DB2. • Cette variable est ignorée si MWTRANSID est utilisé car cette dernière est prioritaire. • Dans tous les cas le paramètre MONITOR est obligatoire et doit correspondre à un programme exécutable sur le serveur CICS.
IXO_SYSTEM_NAME(*)	Nom du serveur pour qui la demande ECI est adressée sur 8 caractères maximum : - facultatif et utilisé avec le protocole CICS/ECI • Si le paramètre IXO_SYSTEMNAME est absent ou non valorisé, le premier nom de serveur défini dans la section Serveur du fichier de configuration CICSCLI.INI est utilisé par défaut. • Si le paramètre est valorisé, le nom du serveur pour qui la demande ECI est adressée doit correspondre à un serveur défini dans la section Serveur du fichier de configuration CICSCLI.INI

(*) : *Disponibles uniquement avec la version 2.5 V08.*

Chaque ligne de fichier ne doit contenir qu'un paramètre.

Exemple de fichier de localisation :

```
[FOCLNT]
<Location1>
COMMENT=moniteur d’exploitation
MONITOR=CLCOMM
LENGTH=8192
MWCODEPAGE=850
MWARE=SOCKET
MWADDRESS=0x00021770C0060A5D
<Location2>
COMMENT=moniteur avec trace
MONITOR=CLCOM2
LENGTH=8192
MWARE=SOCKET
MWADDRESS=0x00021770C0060A5D
<Location3>
COMMENT=moniteur local
MONITOR=CLCOMM
LENGTH=8192
MWARE=LOCAL
```

6.5.3.2. Versions Smalltalk à partir de la 2.5 V08

6.5.3.2.1. Nouvelle structure du fichier VAPLOCAT.INI

Le fichier des localisations est composé d'une liste d'environnements différents dont chaque item est identifié par un nom placé entre <...> (par exemple **<Localhost-Fic-Local>**).

Il est possible pour chaque environnement de définir une liste de paramètres constituée par un ensemble de lignes ayant chacune la syntaxe suivante : PARAMETRE=VALEUR. (une seule définition de paramètre par ligne).

Les paramètres d'un environnement donné sont de deux types : ceux définissant les fonctionnalités générales de la communication (longueur maximale du buffer, moniteur utilisé, etc…) et les paramètres spécifiques à chaque type de communication (par exemple le nom du manager de queue pour MQSERIES ou le nom du fichier de conversion FML pour TUXEDO).

Ce dernier type de paramètres est maintenant préfixé par IXO_ et peut être ajouté à volonté dans une sous-section. Le couple (nom du paramètre, valeur) étant transmis tel quel à la DLL du Middleware, charge à celui-ci de les prendre en compte.

La correspondance des nouveaux noms des paramètres par rapport aux versions antérieures à la 2.5 V08 est donnée dans le tableau suivant (les significations sont bien sûr inchangées) :

Versions antérieures à 2.5 V08	Versions 2.5 V08 et ultérieures
COMMENT	COMMENT
LENGTH	MESSAGE_LENGTH
MONITOR	MONITOR
ADDRESS	LISTENER_ADDRESS
MWARE	COMM_TYPE
MWTIMEOUT	MIDDLEWARE_TIME_OUT
MWCODEPAGE	HOST_CODE_PAGE
MWMAXREPLY	MAX_PENDING_REPLY
MWFMLNAME	IXO_FMLCONVERSIONFILE
MWBUFFERTYPE	Sans objet [1]
MWQUEUEMANAGER	IXO_QUEUEMANAGER
MWREPLYQUEUE	IXO_REPLYQUEUE
MWREPORTQUEUE	Inutilisé
MWREQUESTEXPIRY	IXO_REQUESTEXPIRY
MWREQUESTQUEUE	IXO_REQUESTQUEUE
MWTRANSID	IXO_TRANSID
Inexistant	FOLDER [2]

☞ Le type de buffer en TUXEDO est déterminé par le fait que le paramètre IXO_FMLCONVERSIONFILE est alimenté ou non. Si ce paramètre n'est pas alimenté, alors le type de buffer sera de type CARRAY. Dans le cas contraire, il sera de type FML.

Le paramètre FOLDER permet de spécifier qu'un environnement de localisation sera utilisé par défaut pour le dossier indiqué en valeur de ce paramètre. Cette valeur correspond au nom Pacbase du dossier.

☞ Les fichiers **VAPLOCAT.INI** définis lors de l'utilisation d'une version antérieure à la version 2.5 V08 sont compatibles et peuvent être utilisés tels quels avec la version 2.5 V08.

Si aucun des environnements spécifiés dans le fichier **VAPLOCAT.INI** n'a de paramètre FOLDER (**ce qui est le cas lorsque vous utilisez le fichier d'une version antérieure à la version 2.5 V08**), alors le premier environnement de la liste sera utilisé pour définir la localisation utilisée par défaut par la proxy. Il est fort probable que cet environnement par défaut soit différent de celui qui était sélectionné en utilisant une version antérieure à la version 2.5 V08 υ **il est alors fortement recommandé d'utiliser le paramètre FOLDER afin d'éviter tout problème**.

6.5.3.2.2. Gestion des environnements

Les environnements du fichier **VAPLOCAT.INI** ne sont plus créés ni mis à jour par le générateur de Proxy Vues de Dossier. Les environnements de communication étant spécifiques au poste client, ils sont maintenant gérés en local sur ce poste.

Dans l'attente d'un outil externe indépendant du langage d'implémentation, il est possible de mettre à jour la structure d'un ancien fichier **VAPLOCAT.INI** et de créer, modifier ou supprimer les environnements existant dans ce fichier en utilisant l'éditeur d'environnement **VpcsLocationsEditor** situé dans l'application **VpcsToolsApp**.

6.5.3.2.3. Exemple de fichier de localisations

```
<Location1>
   COMMENT=Moniteur Oracle Socket
   MESSAGE_LENGTH=08192
   MONITOR=MCSKOR
   COMM_TYPE=SOCKET
   MIDDLEWARE_TIME_OUT=120
   LISTENER_ADDRESS=0x0002CACBC0060A5D
   FOLDER=FOCLNA
<Location2>
   COMMENT=Environnement Oracle TUXEDO CARRAY
   MESSAGE_LENGTH=08192
   MONITOR=MCTXOR
   COMM_TYPE=TUXEDO
   FOLDER=FOCLNB
   FOLDER=FOCLNC
<Location3>
   COMMENT=Environnement Oracle TUXEDO FML
   MESSAGE_LENGTH=08192
   MONITOR=MCTXOR
   COMM_TYPE=TUXEDO
   IXO_FMLCONVERSIONFILE=d:\apps\tuxedo\fieldtbl.vap
```

6.5.3.3. Mise en oeuvre

Le fichier **VAPLOCAT.INI** est obligatoire en phase de développement et en phase d'exploitation.

En cas d'absence de ce fichier ou si la syntaxe en est incorrecte, le Gestionnaire d'échanges ne peut être instancié. Pour connaître le paramètre erroné, consultez le fichier trace.

En phase de développement, le générateur de Proxy Vues de Dossier assure la création ou la mise à jour du fichier **VAPLOCAT.INI**.

- S'il n'existe pas, le générateur le crée dans le répertoire courant de VisualAge.

  Pour chaque localisation, les valeurs des paramètres **MONITOR** et **LENGTH** (générés) sont automatiquement positionnées.

  Le développeur a la charge de positionner les autres paramètres.
- S'il existe déjà (dans le répertoire de VisualAge), il est automatiquement chargé. Pour chaque Dossier, le générateur procède alors à une comparaison entre les localisations du fichier d'extraction et les localisations existantes. Il assure la mise à jour du fichier existant :
  - en ajoutant automatiquement les nouvelles localisations,
  - en supprimant automatiquement les localisations qui n'existent plus dans le fichier d'extraction,
  - en modifiant les localisations existantes par la prise en compte des modifications contenues dans le nouveau fichier d'extraction.
    En cas de modification, les paramètres positionnés par le développeur restent inchangés.

A partir de la version 2.5 V08, le fichier **VAPLOCAT.INI** n'est plus mis à jour.

6.6. Test de l'application générée – Packaging

6.6.1. Test de l'application générée

6.6.1.1. Démarrage de l'application

- chemin complet et nom du fichier trace,

Pour démarrer l'application VisualAge, vous codez :

```
abt      -i<image> -v<tracePath>
```

où **<image>** = nom de l'image,

<tracePath> = chemin complet et nom du fichier trace (facultatif).

6.6.1.2. Contrôle des versions

Si l'option du contrôle des versions détecte un déphasage, c'est que le Composant Applicatif et l'objet Proxy n'ont pas été générés avec le même numéro de version. Vous devez donc, selon le cas, :

- regénérer l'objet Proxy si vous avez regénéré uniquement le Composant Applicatif en modifiant sa version,
- ou mettre en exploitation dans VisualAge l'application graphique générée contenant le nouveau composant proxy si cela n'a pas été fait,
- ou mettre en exploitation le Composant Applicatif généré si cela n'a pas été fait.

Au sujet du contrôle des versions, voir également le sous-chapitre **1.2**, ***Compatibilité des Composants Applicatifs /*** Objets Proxy.

6.6.1.3. Aide à la mise au point ou mode trace

Le middleware de Pacbench Client/Serveur est disponible en version optimisée ou en version trace. La dissociation des deux versions est matérialisée par l'extension des fichiers associés aux DLLs du middleware.

Les fichiers d'extension '001' correspondent aux DLLs utilisées en mode TRACE. Le mode TRACE permet d'obtenir un fichier de tous les événements, actions et objets manipulés par le middleware. Ce mode est activé à partir des deux variables d'environnement suivantes :

- **`IXOTRACE`** :

  Cette variable permet d'activer (**`IXOTRACE=1`**) ou de désactiver (**`IXOTRACE=0`**) la trace de l'API Middleware.

- **`IXOTRACE_FILE`** :

  Cette variable permet de spécifier, quand la trace est active (**`IXOTRACE=1`**), le chemin du fichier de trace.
  exemple : *IXOTRACE_FILE=c:\tmp\ixo_err.txt*

Le fichier des événements, actions et objets n'est jamais réinitialisé par les fonctions middleware. Il est donc conseillé de le détruire régulièrement, le middleware se chargeant de le recréer automatiquement.

Les fichiers d'extension '002' correspondent à la version optimisée des DLLs du middleware de Pacbench Client/Serveur. Ces DLLs ne contiennent pas le code permettant de tracer le fonctionnement des communications et n'interprètent donc pas les variables d'environnement réservées à cet effet.

A l'installation de Pacbench Client/Serveur sur la station de développement, les DLLs prêtes à être exécutées correspondent à celles de la version optimisée du middleware.

6.6.2. Packaging

Il existe plusieurs méthodes de packaging, que nous ne décrivons pas dans ce Volume, car elles ne sont pas spécifiques de VisualAge Pacbase.

Pour des informations, reportez-vous au manuel *VisualAge Smalltalk - User's Guide*, partie *Enhancing your applications*, chapitre *Packaging your VisualAge application*.

6.6.2.1. Packaging d'un IC

Si vous disposez d'une version VisualAge Smalltalk 4.02b ou ultérieure, vous pouvez packager le runtime VisualAge Pacbase comme un IC (Image Component).

Pour des explications sur les "Image Components", référez-vous à la documentation *Image Component Developer's Guide and Reference*.

Pour packager le runtime représenté par l'application **`VpcsRuntimeApp`** comme un IC, vous devez effectuer les opérations suivantes :

- Sélectionnez le menu **`Tools`** dans l'Organizer,
- Sélectionnez **`Packaged Images`**,
- Sélectionnez la page **`Instructions In Database`**,
- Sélectionnez l'application **`VpcsToolsApp`** par double-clic,

- Sélectionnez la classe **VpcsRuntimeICPackagingInstructions**,
- Réduisez en cliquant sur le bouton **Reduce**,
- Construisez le fichier du composant en cliquant sur **Output Image**.

L'image est constituée sous le nom **vpcs.ic** . Elle se trouve dans le répertoire de lancement de l'image Smalltalk et peut être employée comme une image réutilisable pour le packaging des applications finales.

6.7. Déploiement de l'application

Pour déployer l'application, suivez les explications indiquées dans la documentation de VisualAge Smalltalk.

Mais vous devez aussi installer, sur le poste de l'utilisateur final, certains fichiers liés à l'utilisation de Pacbench C/S :

- les DLL du middleware,
- **VAPLOCAT.INI**,
- **ABTRULES.NLS** (conversion du code page),
- **VAPERROR.TXT**.

7. Développement d'un Client au standard COM

7.1. Principes généraux

7.1.1. Utilisation des attributs

Un attribut correspond à une information gérée par un objet Proxy. Cette information définit une donnée élémentaire, une liste de données élémentaires ou une liste d'instances de données composées. Un attribut peut correspondre à une constante, à un paramètre ou à un résultat d'action. En fonction du contexte, il est initialisé par l'application graphique ou la Proxy.

On distingue deux types d'attributs :

- ceux qui représentent des variables technologiques. Ils permettent d'affiner le comportement des composants proxy dans VisualAge.
- ceux qui correspondent aux données de la Vue Logique.

La disponibilité d'un attribut est fonction du type de Proxy. Tous les attributs de l'interface publique sont documentés dans le *Manuel de Référence Clients Graphiques : Interface Publique des composants générés*.

7.1.1.1. Contrôles locaux

La Proxy Elémentaire exécute automatiquement les contrôles locaux lors de la création et de la modification d'une instance, via les actions **`createInstance`** et **`modifyInstance`**. Chaque Rubrique appartenant à la Vue Logique est contrôlée.

Les contrôles exécutés sont les suivants :

- Contrôles des listes de valeurs définies dans la description des Rubriques.
- Contrôles des intervalles définis dans la description des Rubriques.
- Contrôles de présence obligatoire définis au niveau de l'appel des Rubriques dans une Vue Logique. La présence des Rubriques de type identifiant ou de type foreign key pour une relation référence de cardinalité minimum de 1 est contrôlée automatiquement.

Si les contrôles locaux détectent une erreur, un message d'erreur est positionné via le Gestionnaire d'erreurs.

Ces contrôles peuvent être déclenchés de manière sélective pour chaque noeud de type racine ou dépendant du Dossier concerné. L'attribut permettant d'activer le contrôle des Rubriques sur le serveur est **`serverCheckOption`**.

En revanche, la Proxy n'assure pas les contrôles de numéricité et de date, qui doivent être pris en charge par l'application graphique.

7.1.1.2. Gestion des sous-schémas

Les sous-schémas spécifiés dans la description de la Vue Logique peuvent être pris en compte lors des actions de sélection/lecture si les Composants Applicatifs gèrent la présence des Rubriques (options **VECTPRES=YES** ou **CHECKSER=YES**).

Tous les nœuds disposant d'au moins un sous-schéma possèdent les attributs suivants :

- **subSchema**, qui permet d'attribuer le sous-schéma désiré lors d'une action de sélection/lecture du Composant Applicatif du nœud. Cet attribut peut être alimenté à partir des attributs suivants. Il est ignoré pour toute action de mise à jour.
- **getSubSchemaCount** et **getSubSchemasElementAt(index)**, qui permet de lister tous les sous-schémas disponibles pour le nœud. Comme il n'est pas possible dans le Référentiel VisualAge Pacbase d'affecter un nom au sous-schéma, chaque sous-schéma est désigné par **SubSchema<n>** (avec **n** de 01 à 10).

7.1.1.3. Contrôle de longueur des champs de l'attribut detail

Pour chaque champ de l'attribut **detail**, lors des contrôles effectués pour une création ou une modification locale d'instance, la longueur de la valeur contenue dans l'attribut ne doit pas dépasser la longueur maximale de la valeur de cet attribut.

Le contrôle s'effectue systématiquement (sauf si l'attribut n'appartient pas au sous-schéma courant) même si l'attribut a été défini comme "non à contrôler dans le client".

Une erreur locale est envoyée en cas de longueur excessive.

☞ Il n'y a pas de contrôle de longueur sur les champs inclus dans les buffers utilisateur. Si la longueur est excessive, elle est tronquée à la longueur maximale.

7.1.1.4. Sélection du tri local ou serveur sur une liste d'instances

L'attribut **localSort** permet de spécifier si la Proxy trie les instances de l'attribut **rows** à partir du tri défini en local (**true**) ou laisse les instances triées dans l'ordre d'envoi du serveur (**false**).

Vous pouvez à tout moment modifier le type de tri.

☞ Cet attribut n'est pas effectif dans les services utilisateur.

7.1.1.4.1. Tri local

Le tri local des instances de l'attribut **rows** correspond au fonctionnement standard de la Proxy si le paramétrage n'a pas été modifié après sa génération. Dans ce contexte, deux types de tri sont possibles :

- Si aucun critère de tri n'est défini en local, la Proxy trie implicitement les instances dans l'ordre croissant des identifiants définis sur la Vue Logique.
- Si un critère de tri est défini en local, la Proxy trie les instances dans l'ordre défini par ce dernier.

Dans tous les cas, la création locale d'une instance insère celle-ci en fonction du critère de tri courant appliqué dans l'attribut **rows**.

Le changement dynamique ou la suppression du critère de tri local induit immédiatement un tri sur les instances contenues dans l'attribut **rows**.

7.1.1.4.2. Tri serveur

Le tri serveur des instances de l'attribut **rows** est déclenché si l'attribut **localSort** est à **false** ou si le paramétrage standard du nœud concerné a été modifié. Dans ce contexte, les instances contenues dans l'attribut **rows** sont présentées dans l'ordre dans lequel elles ont été reçues du serveur, quel que soit le contenu du critère de tri défini en local.

Dans le contexte de gestion manuelle des collections ou de pagination en mode extend, les instances reçues sont ajoutées en fin de la collection existante dans l'attribut **rows**.

Toute instance créée localement est systématiquement ajoutée à la fin de la collection existante dans l'attribut **rows**. Dans ce contexte, une instance qui n'est pas positionnée en fin d'une collection qui est supprimée et recréée localement est transférée à la fin de la collection contenue dans l'attribut **rows**.

7.1.2. Utilisation des actions

7.1.2.1. Mise en œuvre

Une action correspond à un traitement qu'un objet Proxy peut exécuter. Elle est déclenchée à l'aide d'une connexion entre un événement de l'application graphique et le code d'une action d'une Proxy.

La disponibilité d'une action est fonction du type de Proxy. Toutes les actions de l'interface publique sont documentées dans le manuel *Manuel de Référence Clients Graphiques : Interface Publique des composants générés*.

7.1.2.2. Les différents types d'actions serveur

Les actions serveur exécutent des traitements implémentés dans un ou plusieurs Composants Applicatifs associés au Dossier. Ces actions émettent une requête vers les Composants Applicatifs qui renvoient un résultat sur la station de travail. Les requêtes et les réponses contiennent généralement des paramètres techniques, des instances de Vue Logique associées à un ou plusieurs noeuds et des informations contextuelles définies dans un buffer utilisateur.

Il faut distinguer deux types d'actions serveur.

- celles qui accèdent systématiquement au serveur :
  - **selectInstances**,
  - **readInstance**,
  - **readInstanceAndLock**,
  - **readWithFirstChildren**,
  - **readWithAllChildren**,
  - **readWithFirstChildrenAndLock**,
  - **readWithAllChildrenAndLock**,
  - **readAllChildrenFromDetail**,
  - **readWithAllChildrenFrom**.
- celles qui ne font pas un accès systématique au serveur :
  - **readNextPage** : il y a accès au serveur sauf si lors de la précédente sélection, l'événement **NO_PAGE_AFTER** a été renvoyé.
  - **readPreviousPage** : il y a accès au serveur sauf si lors de la précédente sélection, l'événement **NO_PAGE_BEFORE** a été renvoyé.

- **readFirstChildrenFromDetail** : il y a accès au serveur, sauf si l'attribut **maxNumberOfRequestedInstances** des Proxy Dépendantes est positionné à 0 et si l'attribut **globalSelection** est positionné à false.
- **readWithFirstChildrenFrom** : il y a accès au serveur, sauf si l'attribut **maxNumberOfRequestedInstances** des Proxy Dépendantes est positionné à 0 et si l'attribut **globalSelection** est positionné à false.
- **checkExistenceOfDependencies** : il y a accès au serveur sauf s'il existe en local la possibilité de vérifier l'existence d'instances dépendantes.
- **updateFolder** : il n'y a d'accès au serveur que s'il existe au moins une instance du noeud concerné modifiée dans son attribut **updatedFolders**.

7.1.2.3. Gestion des lectures d'un Dossier

La lecture massive de la racine d'un Dossier permet de lire sur un composant client toutes les occurrences du noeud racine du Dossier présentes dans la base de données. Les actions concernées sont **selectInstances** et **readNextPage**.

7.1.2.3.1. Lecture massive par anticipation des noeuds dépendants

Dans la cinématique des architectures client/serveur, une application graphique manipulant un Dossier cherche à acquérir des informations par anticipation pour minimiser les échanges avec les serveurs.

Dans un réseau hiérarchique, plusieurs actions d'anticipation des lectures peuvent être envisagées :

- La première action de type '**allChildren**' lit *toutes* les instances dépendantes de l'instance sélectionnée dans l'attribut **detail** de la Proxy Elémentaire parente.
- La deuxième action de type '**firstChildren**' ne lit que les instances *immédiatement* dépendantes de l'instance sélectionnée dans l'attribut **detail** de la Proxy Elémentaire parente.

La première action de lecture massive par anticipation n'est disponible que sur la Proxy Racine.

La deuxième action de lecture massive par anticipation est disponible sur la Proxy Racine ou sur les Proxy Dépendantes qui possèdent également des Proxy Dépendantes.

7.1.2.3.2. Transfert d'instance entre les attributs rows et detail

L'attribut **rows** n'est pas accessible directement mais par l'intermédiaire des actions **getRowsCount()** et **getRowsElementAt(Int i)**.

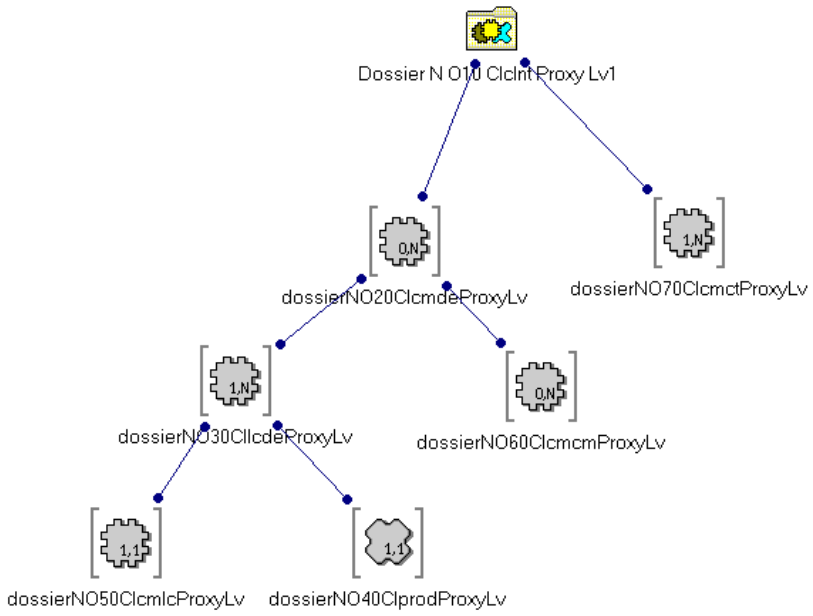
Le transfert d'instance entre l'attribut **rows** et **detail** permet d'alimenter l'attribut **detail** avec une instance initialement lue par une action de lecture massive.

Ce transfert n'est disponible que sur les Proxy Racine et Dépendante. Il correspond à une action de lecture locale qui alimente également toutes les instances locales des Proxy Dépendantes connues par la Proxy Vue de Dossier. Le transfert est réalisé par l'intermédiaire de l'action **`getDetailFromData`**.

7.1.2.3.3. Lecture massive et transfert d'instance entre les propriétés rows et detail : cinématique de fonctionnement

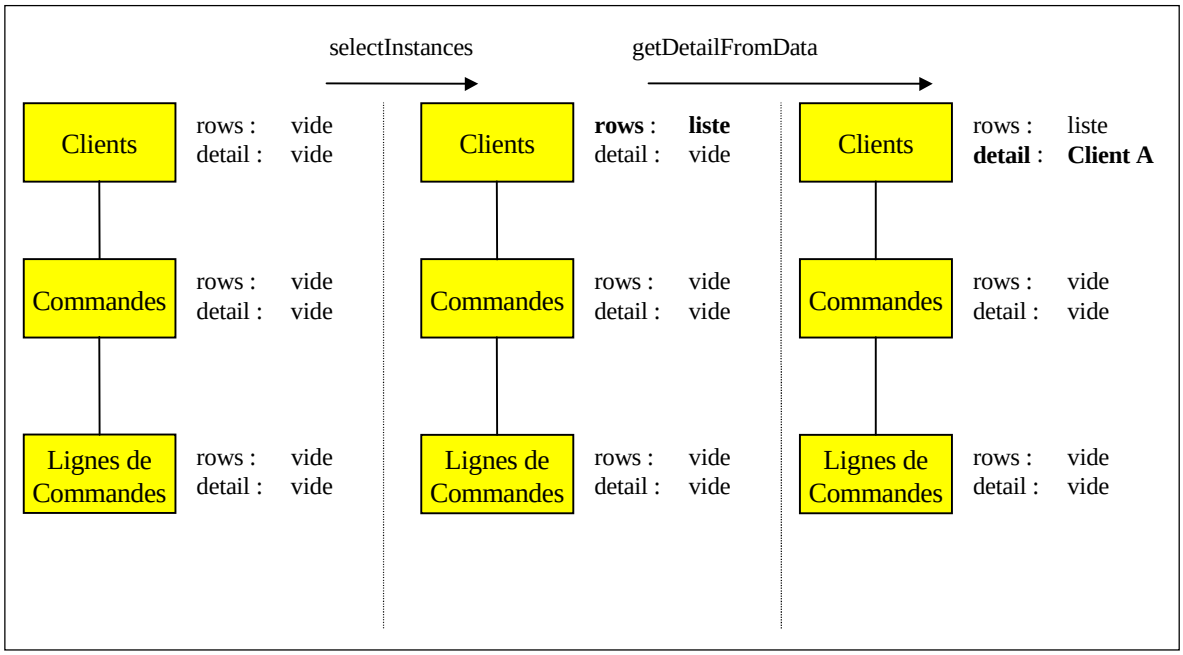
Cet exemple illustre l'alimentation de **`detail`** avec une instance préalablement transférée dans **`rows`** par une action de lecture massive d'un noeud racine ou dépendant et le principe de la lecture massive par anticipation des noeuds dépendants.

Il est basé sur la Proxy Vue de Dossier générée dans le chapitre précédent dont nous rappelons la composition en Proxy Elémentaires dans l'arborescence ci-dessous :



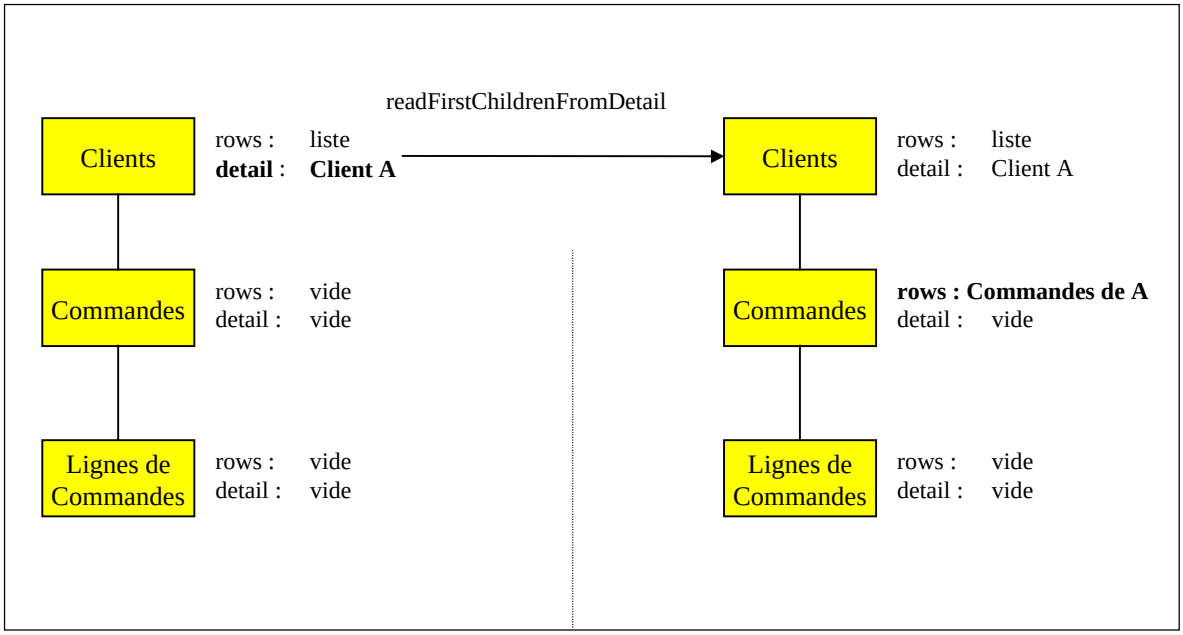
Pour les besoins de notre exemple, nous utilisons ici trois Proxy Elémentaires :

- la Proxy Racine **`dossierNO10ClclntProxyLv`** correspondant au noeud **`Clients`** qui gère les clients dans le système d'information décrit par le Dossier.
- la Proxy Dépendante **`dossierNO20ClcmdeProxyLv`** correspondant au noeud **`Commandes`** qui gère les commandes dans le système d'information décrit par le Dossier.
- la Proxy Dépendante **`dossierNO30CllcdeProxyLv`** correspondant au noeud **`Lignes de Commandes`** qui gère les lignes de commandes dans le système d'information décrit par le Dossier.



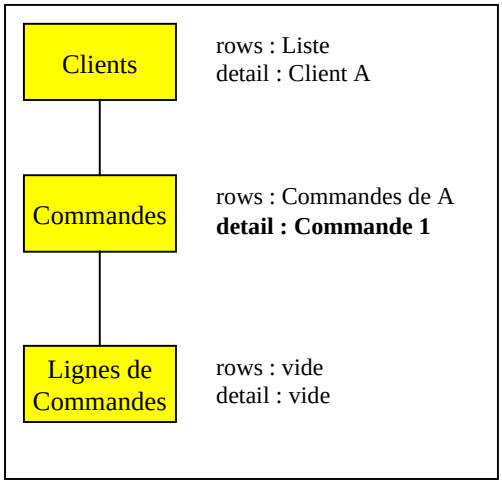
L'attribut **`detail`** du noeud Clients contient à présent le Client A. Ensuite, pour lire les instances dépendantes du Client A, c'est-à-dire ses commandes, vous disposez de trois solutions.

SOLUTION 1



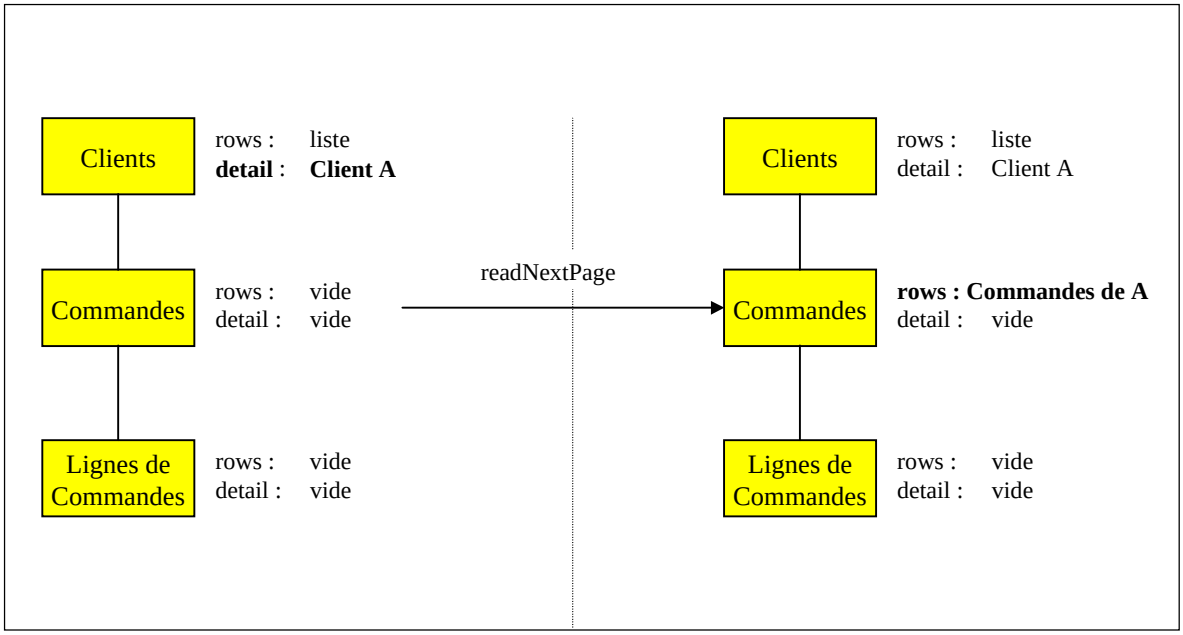
L'action **readFirstChildrenFromDetail** sur la Proxy Racine Clients alimente non seulement l'attribut **rows** de la Proxy Dépendante Commandes pour le Client A mais aussi l'attribut **rows** des éventuelles autres Proxy Elémentaires directement dépendantes de la Proxy Racine.

Dans ce contexte, l'action **getDetailFromData** sur la Proxy Dépendante Commandes provoque :



Pour lire ensuite les lignes de commandes de la Commande 1 du Client A, utilisez l'action **readFirstChildrenFromDetail** sur Commandes ou l'action **readNextPage** sur Lignes de commandes.

SOLUTION 2

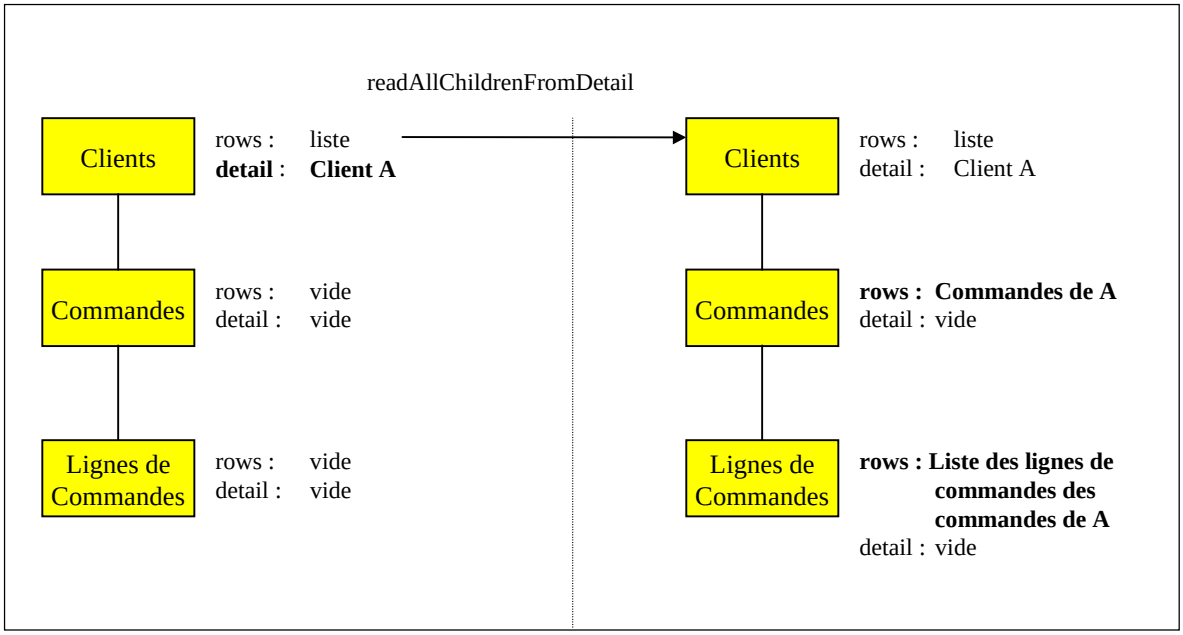


L'action **`readNextPage`** sur la Proxy Dépendante Commandes alimente son attribut **`rows`**. Les instances des éventuelles autres Proxy Elémentaires directement dépendantes de la Proxy Racine ne sont pas lues.

Dans ce contexte, l'action **`getDetailFromData`** sur la Proxy Dépendante Commandes provoque le même résultat que dans la solution 1.

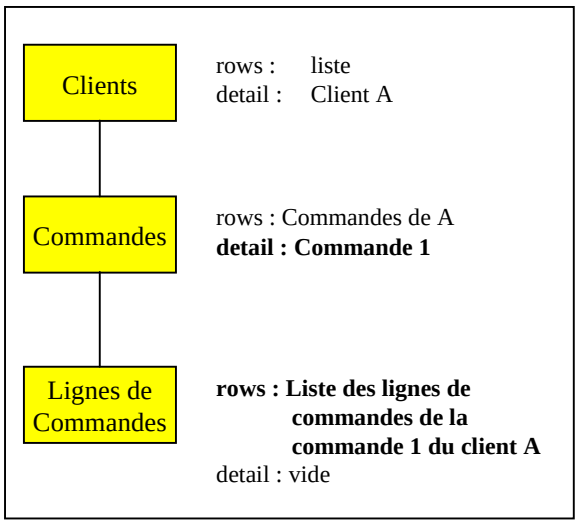
Pour lire ensuite les lignes de commandes de la Commande 1 du Client A, il faut procéder de la même manière qua dans la solution précédente.

SOLUTION 3



L'action **`readAllChildrenFromDetail`** sur la Proxy Racine Clients lit non seulement toutes les commandes de A mais aussi toutes les éventuelles autres instances dépendantes de A quelque soit leur niveau hiérarchique : ainsi dans notre exemple, toutes les lignes de commandes de toutes les commandes de A sont lues.

Dans ce contexte, l'action **`getDetailFromData`** sur la Proxy Dépendante Commandes provoque :



L'action **`getDetailFromData`** alimente automatiquement l'attribut **`rows`** de la Proxy Dépendante Lignes de commandes avec les lignes de commandes de la Commande 1 du Client A, ces lignes ayant été transférées en local au préalable par l'action **`readAllChildrenFromDetail`**.

7.1.2.3.4. Lecture massive des noeuds références

La lecture d'un noeud référence est considérée comme une aide sur critères. Elle permet de montrer à l'utilisateur final une liste d'informations potentiellement référençables sur un noeud dépendant.

Les informations présentées à l'utilisateur correspondent aux informations nécessaires et suffisantes pour assister l'utilisateur final dans son choix.

Pour optimiser le volume des caractères transmis pour ce type de service, les Vues Logiques disposent d'un sous-schéma de type « aide sur critères » qui permet de sélectionner les Rubriques concernées au niveau du serveur.

Les lectures massives des noeuds références sont exécutées sur demande et ne peuvent pas être intégrées dans le traitement des lectures par anticipation. Les actions concernées sont `selectInstances` et `readNextPage`.

7.1.2.3.5. Principe de pagination dans les noeuds d'un Dossier

Deux types de pagination sont proposés sur les noeuds d'un Dossier :

- Le premier type de pagination, appelé pagination en mode *non-extend*, permet de paginer en avant et en arrière sur une collection prédéfinie, par l'intermédiaire d'actions spécifiques. Chaque action exécute une demande de lecture au serveur et son résultat remplace le résultat de la lecture précédente. Ce type de pagination n'est applicable que sur les noeuds racines ou références.
- Le deuxième type de pagination, appelé pagination en mode *extend*, permet de lire graduellement les instances d'une collection définie, au fur et à mesure des demandes de lectures des pages suivantes. Dans ce contexte, la lecture des pages précédentes disparaît et cette fonction est reprise en local par les ascenseurs du contrôle graphique qui présente la liste des instances. Ce type de pagination est applicable à tous les noeuds d'un Dossier.

7.1.2.3.6. Critères de sélection associés aux lectures massives

Les critères de sélection associés aux lectures massives sont des attributs élémentaires ou composés associés à chaque noeud d'un Dossier. Ils se répartissent en deux types :

- Les critères de sélection fonctionnels correspondent à l'identifiant et aux éléments nécessaires à la définition des méthodes d'extraction de la Vue Logique associée au noeud. Ces critères sont les suivants :
  - l'attribut `selectionCriteria` définit les Rubriques de type identifiant et paramètres par valeur.
  - l'attribut `extractMethodCode` définit le code de la méthode d'extraction souhaitée.
  - les actions `getExtractMethodCodesCount()` et `getExtractMethodCodesElementAt(Int i)` permettent d'accéder à la liste des méthodes d'extraction disponibles.
- Les critères de sélection organiques correspondent aux informations permettant de maîtriser le volume des instances sélectionné sur chaque noeud.
  - `globalSelection` est un attribut booléen qui, lorsqu'il est positionné à true, permet de lire la totalité des instances du noeud sur une requête de sélection.

- **maxNumberOfRequestedInstances** est un attribut numérique qui indique, lorsque l'attribut **globalSelection** est à false, le nombre d'instances d'un noeud à lire sur une requête de sélection. Cet attribut peut contenir la valeur 0. Dans ce cas, la branche n'est pas lue lors d'une lecture massive par anticipation.

7.1.2.3.7. Limitation du champ des lectures massives

Lors d'une lecture massive avec utilisation de critères de sélection, seules les instances qui correspondent à ces critères sont lues.

Cependant, dans le cas d'une lecture massive commandée par une lecture par anticipation de type **allChildren**, on considère que l'atribut **globalSelection** est positionné à true sur chaque noeud.

7.1.2.3.8. Lecture d'une instance d'un noeud racine ou dépendant

La lecture d'une instance d'un noeud racine ou dépendant permet d'alimenter directement l'attribut **detail** d'un noeud sans passer au préalable par une sélection massive d'une collection des instances de ce noeud.

Ce type de lecture est considéré comme une sélection de collection et annule donc la sélection précédente même si celle-ci correspondait à une lecture massive. L'alimentation de l'attribut **detail** entraîne donc l'initialisation de l'attribut **rows**.

Les actions qui implémentent cette fonction de sélection permettent :

- Une lecture de l'instance sans ses dépendances (**readInstance**).
- Une lecture de l'instance avec ses dépendances de premier niveau (**readWithFirstChildren**).
- Une lecture de l'instance avec toutes ses dépendances (**readWithAllChildren**).

7.1.2.3.9. Lecture d'une instance d'un noeud référence

La lecture d'une instance d'un noeud référence ne peut pas activer de processus de lecture massive par anticipation. Elle permet de lire la totalité de la description de l'instance du noeud appelé dans son attribut **detail**.

Seule l'action **readInstance** est donc disponible sur un noeud référence.

Les noeuds références sont dépourvus d'action de mise à jour. Leurs attributs **rows** et **detail** sont indépendants. Le résultat de l'action **readInstance** n'initialise donc pas l'attribut **rows**.

L'attribut **rows** permet de visualiser les informations suffisantes pour permettre d'affecter une des instances du noeud référence à l'instance du noeud référençant.

L'attribut **rows** n'est pas accessible directement mais par l'intermédiaire de deux actions **getRowsCount()** et **getRowsElementAt(Int i)**.

L'attribut **detail** permet de consulter la totalité de la description d'une instance référencée.

La structure de ces deux attributs peut donc être différente.

7.1.2.3.10. Critères de sélection associés aux lectures d'une instance

L'identifiant de l'instance du noeud à lire est spécifié dans les critères de sélection fonctionnels associés au noeud.

Pour les noeuds dépendants, l'identifiant du noeud correspond à l'identifiant de la Vue Logique associée au noeud débarrassé des Rubriques définissant les identifiants des Vues Logiques hiérarchiquement supérieures. Ces identifiants hiérarchiques sont initialisés automatiquement par la Proxy Vue de Dossier en fonction de la navigation dans le Dossier.

7.1.2.4. Gestion des mises à jour d'un Dossier

7.1.2.4.1. Mises à jour locales

Les services de mise à jour locales sont disponibles sur chaque noeud de type racine ou dépendant du Dossier.

- Création d'une instance de noeud,
- Modification d'une instance de noeud,
- Annulation d'une instance de noeud.

Cependant, il existe d'autres règles inhérentes à la gestion des Dossiers :

- La création d'une instance d'un noeud dépendant n'est autorisée que si la hiérarchie des instances contenues dans les attributs `detail` des noeuds supérieurs existe.
- L'annulation d'une instance d'un noeud entraîne l'annulation récursive des instances locales des noeuds dépendants.

Pour permettre au développeur de gérer des messages permettant d'avertir l'utilisateur de l'impact d'une annulation en cascade, une action de vérification d'existence d'instances dépendantes est disponible sur les noeuds de type racine ou dépendant.

Cette action (`checkExistenceOfDependencies`) émet un résultat booléen true ou false. Si aucune dépendance n'est trouvée dans le cache local du Dossier et que l'instance concernée n'a pas été créée localement, elle émet une requête de vérification sur le serveur.

Pour qu'un utilisateur puisse revenir en arrière sur les manipulations locales d'une instance de Dossier, une action `undoAllLocalFolderUpdates` permet d'éliminer toutes les mises à jour locales sur tous les noeuds du Dossier appliquées depuis la dernière exécution d'une action de mise à jour serveur. Cette action n'est disponible que sur le noeud racine du Dossier. Une autre action `undoLocalFolderUpdates` permet d'éliminer toutes les mises à jour associées à l'instance passée en paramètre.

7.1.2.4.2. Mises à jour serveur

La Proxy Racine est seule à disposer des actions de mise à jour serveur.

Les mises à jour serveur correspondent aux actions permettant à un composant client d'envoyer au serveur l'ensemble des mises à jour locales effectuées depuis la dernière action de mise à jour de ce type.

Ces mises à jour concernent toutes les instances dépendantes modifiées. Elles peuvent concerner plusieurs instances de Dossier.

Lorsqu'une action de mise à jour serveur renvoie des erreurs, elle laisse le Dossier en statut « modifié localement » et seule la correction des erreurs et le renvoi des mises à jour, ou de l'action **undoAllLocalFolderUpdates** ou **undoLocalUpdates** permettent d'effectuer de nouvelles sélections de collection.

Les actions de mise à jour serveur exécutent, avant l'émission de la requête sur le serveur, un contrôle de l'intégrité du Dossier pour les instances créées localement. Cette action contrôle, pour chaque instance de noeud créé en local, les cardinalités minimum de chaque lien et émet une erreur si le nombre des instances dépendantes ne respecte pas les propriétés des liens associés.

Une action de mise à jour serveur peut être accompagnée d'une demande de rafraîchissement des instances mises à jour dans le cas où certaines Rubriques de ces instances, comme les identifiants, sont calculées par le serveur. Cette demande de rafraîchissement passe par l'utilisation de l'attribut **refreshOption**.

7.1.2.4.3. Gestion des mouvements utiles

La gestion des mouvements utiles, assurée par le cache local, est entièrement automatique et transparente pour le développeur.

Elle 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. Si plusieurs mises à jour locales ont été effectuées sur une même instance de noeud, seule la dernière sera envoyée au serveur.

7.1.2.4.4. Changement de sélection d'une collection

L'événement **ABOUT_TO_CHANGE_SELECTION** est envoyé au Client par la Proxy Racine lorsqu'une action de sélection serveur sur un noeud du Dossier remplace sa collection locale courante et que cette collection ou les collections dépendantes contiennent des modifications locales candidates à une mise à jour serveur. Si l'événement n'est pas intercepté lors de la programmation, les mises à jour sont perdues.

7.1.2.4.5. Réinitialisation des instances du cache local

L'action **resetCollection** permet d'éliminer explicitement toutes les instances contenues dans le cache d'une Proxy Vue de Dossier avant de reconstituer une nouvelle collection. Cette action est disponible sur tous les types de nœuds d'une Proxy Vue de Dossier disposant de l'attribut **rows**.

L'attribut **rows** n'est pas accessible directement mais par l'intermédiaire de deux actions **getRowsCount()** et **getRowsElementAt(Int i)**.

7.1.2.4.6. Gestion des collections d'instances

La gestion des collections d'instances peut se faire soit automatiquement, soit manuellement par le positionnement de l'attribut booléen **manualCollectionReset** disponible sur tous les types de nœuds d'une Proxy Vue de Dossier. Le mode de gestion manuel permet de constituer des collections hétérogènes par une succession d'actions de sélection de collections et de pagination.

7.1.2.5. Services utilisateur

Chaque noeud de type racine ou dépendant dispose pour mettre en oeuvre un service utilisateur des éléments suivants :

- Deux actions permettant d'obtenir la liste des services utilisateur disponibles sur ce nœud : **`getUserServiceCodesCount()`** et **`getUserServiceCodesElementAt(Int i)`**.
- Deux actions permettant de stocker localement des instances de Vue Logique à traiter pour le prochain service utilisateur : **`getUserInputRowsCount()`** et **`getUserInputRowsElementAt(Int i)`**.
- Deux actions permettant de présenter plusieurs instances de Vue Logique renvoyées par un service utilisateur : **`getUserOutputRowsCount()`** et **`getUserOutputRowsElementAt(Int i)`**.
- Un attribut présentant les instances de Vue Logique candidates à l'exécution du prochain service utilisateur. Cet attribut est **`userDetail`**.
- Des actions locales permettant de mémoriser chaque instance de Vue Logique à envoyer au serveur pour l'exécution d'un service utilisateur : **`createUserInstance`**, **`modifyUserInstance`** et **`deleteUserInstance`**.

Le noeud racine du Dossier dispose en plus des éléments suivants :

- Une action permettant d'exécuter l'ensemble des services utilisateur paramétrés sur chaque noeud du Dossier. Cette action est **`executeUserService`**.
- Une action permettant d'effacer toutes les instances locales mémorisées pour tous les noeuds du Dossier. Cette action est **`resetUserRows`**.
- Une action permettant d'effacer l'instance courante mémorisée en local. Cette action est **`resetUserServiceCodes`**.

Ce principe permet d'exécuter dans la même requête un ensemble de 1 à n services utilisateur répartis sur des noeuds différents. L'ordre d'exécution de ces services correspond à l'ordre hiérarchique des noeuds, en parcourant l'arbre de haut en bas et de gauche à droite.

7.1.2.6. Verrouillage logique de la base

Les mécanismes d'upload-download associés à un Dossier augmentent le temps écoulé entre la lecture d'un Dossier et l'affichage de la mise à jour.

Dans ce contexte, sans mécanisme de verrouillage, deux utilisateurs peuvent modifier la même instance de Dossier. Les mises à jour cumulées deviennent difficiles à gérer.

Pour permettre à l'utilisateur d'utiliser un Dossier dans un mode d'appropriation exclusif, deux types de verrouillage d'une instance de noeud sont à sa disposition :

- Le verrouillage optimiste qui fonctionne sur le principe de la vérification de l'évolution d'un **`TimeStamp`** avant d'exécuter le traitement de mise à jour.
- Le verrouillage pessimiste qui s'approprie en mise à jour exclusive une entité au moyen de l'enregistrement d'une ressource spécifique. Dans ce cas, le traitement de mise à jour du Dossier est effectué avant de libérer la ressource exclusive.

Le traitement de verrouillage serveur est déclenché en fonction de l'exécution explicite d'une action spécifique disponible sur la Proxy Racine. Cette action est **`lock`**.

Le traitement de déverrouillage serveur est déclenché automatiquement avec l'exécution d'une action de mise à jour serveur ou explicitement en fonction de l'exécution d'une action spécifique disponible sur la Proxy Racine. Cette action spécifique est **`unLock`**.

L'écriture du traitement de verrouillage dans le Composant Applicatif racine est à la charge du développeur.

Ce traitement reçoit l'identifiant de l'instance de Vue Logique à verrouiller ainsi que le type de demande (verrouillage ou déverrouillage) à exécuter.

Il doit positionner en retour un statut permettant d'accorder ou de refuser le verrouillage ainsi que le **`TimeStamp`** ou le nom de la ressource utilisée.

En cas de refus de verrouillage, la Proxy Racine émet un événement **`LOCK_FAILED`** et toutes les actions de mise à jour locales ou serveur exécutées ultérieurement sont invalidées pour l'instance de Dossier spécifiée. Dans ce cas, le Dossier passe en statut « lecture uniquement ».

Le verrouillage logique est défini dans l'entité Dossier ou dans le Composant Applicatif en cas de développement mono-vue.

Lorsque l'option de verrouillage logique est active sur un Dossier, toute demande de lecture de l'attribut **`detail`** de la Proxy Racine peut être accompagné d'une demande de blocage logique au niveau du serveur.

7.1.2.7. Gestion de la présence des Rubriques

Les deux actions suivantes vous permettent de gérer la présence des Rubriques de la Vue Logique au niveau de la Proxy.

L'action **`is<corub>Present`** permet de tester la présence ou l'absence de la Rubrique **`corub`**. Elle est générée pour toutes les classes **`DataDescription`** et **`UserDataDescription`**.

L'action **`set<corub>Present(aBoolean)`** permet d'indiquer la présence ou l'absence de la Rubrique avant toute action de mise à jour locale. Elle est générée pour toutes les classes **`DataDescription`** et **`UserDataDescription`**.

Par défaut, les Rubriques sont considérées comme étant absentes, sauf si une valeur par défaut et/ou une liste de valeurs ont été indiquées dans la description VisualAge Pacbase.

7.1.2.8. Gestion du contrôle des Rubriques

Les deux actions suivantes vous permettent de gérer le contrôle des Rubriques de la Vue Logique au niveau de la Proxy.

L'action **`setCheck<fieldIndex,aBoolean>`** permet d'activer ou d'inhiber les contrôles serveur sur une Rubrique avant toute action de mise à jour locale.

L'action **`getCheck<fieldIndex>`** permet de tester si les contrôles serveur sont activés pour une Rubrique.

Ces deux actions sont générées pour toutes les classes **`DataDescription`** des nœuds racines ou dépendants dont le Composant Applicatif possède les options **`NULLMNGT=YES`** et **`CHECKSER=YES`** et un service de mise à jour.

Par défaut, toutes les Rubriques sont considérées à contrôler (si l'attribut **`serverCheckOption`** est positionné à **`true`**).

7.1.2.9. Gestion des sous-schémas

Toute action serveur de sélection et de lecture prend en compte le sous-schéma présent dans l'attribut **`subSchema`** et retourne donc les valeurs des Rubriques du sous-schéma. Si une action de sélection est suivie d'une action de pagination, le sous-schéma pris en compte est celui associé à l'action de sélection.

Toute action de création locale est effectuée hors sous-schéma.

Toute action de modification/suppression locale est effectuée sur le sous-schéma associé à l'instance, c'est à dire :

- si la modification/suppression est effectuée sur une instance créée localement, le sous-schéma est vide.
- si la modification/suppression est effectuée sur une instance lue, le sous-schéma est celui associé à la sélection de cette instance.

De plus, les deux actions suivantes sont spécifiques à la gestion des sous-schémas.

L'action **`resetSubSchema`** permet de réinitialiser l'attribut **`subSchema`** à vide, c'est à dire de ne sélectionner aucun sous-schéma.

L'action **`completeInstance`** permet de récupérer, par appel du Composant Applicatif associé à la Vue Logique, les valeurs des Rubriques n'appartenant pas au sous-schéma.

L'action **`belongsToSubschema`** permet de savoir si la Rubrique passée en paramètre appartient au sous-schéma associé à l'instance contenue dans l'attribut **`detail`**.

7.1.3. Utilisation des événements

Les événements émis par une Proxy permettent de déclencher des actions applicatives appartenant à l'application graphique. Ces traitements sont exécutés en connectant un événement de la Proxy à une ou plusieurs actions de l'application graphique. L'exécution conditionnelle des actions est facilitée par le fait qu'un événement est toujours accompagné de son événement contraire, les deux événements ne pouvant jamais être émis simultanément. L'émission d'un événement entraînant le stockage de ce dernier dans une pile, il est nécessaire que l'application graphique accède à cette dernière régulièrement à l'aide des actions **`getServerEventsCount`** et **`popServerEvent`**.

La disponibilité d'un événement est fonction du type de Proxy. Tous les événements de l'interface publique sont documentés dans le *Manuel de Référence Clients Graphiques : Interface Publique des composants générés*.

7.1.3.1. Gestion événementielle des lectures massives

La gestion événementielle des lectures massives permet au développeur d'avoir des informations sur l'état de la collection d'instances contenu dans un noeud. Chaque action de pagination disponible propose son propre système de pagination événementielle.

L'action de pagination en mode non-extend peut émettre les quatre événements suivants :

- **NO_PAGE_BEFORE** : Cet événement est émis par un noeud racine ou référence à la fin de l'exécution d'une action de sélection de collection ou de pagination lorsque celle-ci ne renvoie pas d'erreur et que la page lue est la première de la collection courante.

- **PAGE_BEFORE** : Cet événement est émis par un noeud racine ou référence à la fin de l'exécution d'une action de sélection de collection ou de pagination lorsque celle-ci ne renvoie pas d'erreur et que la page lue n'est pas la première de la collection courante.

- **NO_PAGE_AFTER** : Cet événement est émis par un noeud racine ou référence à la fin de l'exécution d'une action de sélection de collection ou de pagination lorsque celle-ci ne renvoie pas d'erreur et que la page lue est la dernière de la collection courante.

- **PAGE_AFTER** : Cet événement est émis par un noeud racine ou référence à la fin de l'exécution d'une action de sélection de collection ou de pagination lorsque celle-ci ne renvoie pas d'erreur et que la page lue n'est pas la dernière de la collection courante.

L'action de pagination en mode extend peut émettre les deux événements suivants :

- **PAGE_AFTER** : Cet événement est émis par tout type de noeud à la fin de l'exécution d'une action de sélection de collection ou de pagination avant lorsque celle-ci ne renvoie pas d'erreur et que le nombre d'instances contenu dans le noeud ne correspond pas au nombre d'instances total contenu dans la base.

- **NO_PAGE_AFTER** : Cet événement est émis par tout type de noeud à la fin de l'exécution d'une action de sélection de collection ou de pagination avant lorsque celle-ci ne renvoie pas d'erreur et que le nombre d'instances contenu dans le noeud correspond au nombre d'instances total contenu dans la base au moment de la requête.

7.1.3.2. Gestion événementielle des lectures d'une instance

Une Vue Logique peut être mappée sur une ou plusieurs entités de stockage physique. Dans ce contexte, la gestion événementielle de la lecture d'une instance de Vue Logique peut émettre deux événements :

- **NOT_FOUND** lorsque l'instance recherchée n'existe pas dans la base de données. Cet événement peut être émis lorsque la Vue Logique est mappée sur une ou plusieurs tables.

- **NOT_COMPLETE** lorsque l'instance recherchée est incomplète. C'est-à-dire qu'au moins une table a satisfait au critère de sélection et au moins une table n'a pas satisfait à ce critère. Cet événement ne peut être émis que dans un contexte de multi-mapping.

7.2. Gestion des erreurs

Il existe quatre types d'erreurs :

- Les erreurs locales, émises par le Composant Client, qui correspondent à des erreurs de manipulation ou de saisie dans le Composant Client,
- Les erreurs serveur, envoyées par le Composant Applicatif, qui sont des erreurs d'accès aux données et des erreurs utilisateur positionnées dans le serveur,
- Les erreurs système, envoyées aussi par le Composant Applicatif, qui correspondent à un déphasage entre la Proxy et les Composants Applicatifs,
- Les erreurs de communication.

7.2.1. Erreurs locales

Les messages d'erreur locaux sont listés ci dessous :

- ASYNCHRONOUS_VIOLATION

  Cette erreur se produit en cas de demande de verrouillage, de déverrouillage ou de vérification d'existence d'instances dépendantes en mode asynchrone.
- CARDINALITY_VIOLATION

  Cette erreur se produit en cas de non respect des cardinalités lors de l'activation d'une méthode de mise à jour.
- CURRENT_INSTANCE_MISSING

  Cette erreur se produit quand une méthode est appliquée à la propriété `detail` alors que cette propriété ne contient pas d'instance.
- CURRENT_USER_INSTANCE_MISSING

  Cette erreur se produit quand une méthode utilisateur est appliquée à la propriété `userDetail` alors que cette propriété ne contient pas d'instance.
- FOLDER_USER_CONTEXT_LENGTH_ERROR

  Cette erreur se produit quand le contenu d'une rubrique du buffer utilisateur a une taille supérieure à la taille autorisée pour la rubrique.
- INSTANCE_ALREADY_LOCKED

  Cette erreur se produit en cas de demande de verrouillage d'une instance déjà verrouillée sur le serveur.
- INSTANCE_NOT_LOCKED

  Cette erreur se produit en cas de demande de déverrouillage sur une instance non verrouillée sur le serveur.
- INVALID_CHANGE

  Cette erreur se produit lorsque l'instance à modifier n'existe pas dans le cache local.
- INVALID_CREATION

  Cette erreur se produit en cas de création d'une instance qui existe déjà dans le cache local.
- INVALID_DELETION

  Cette erreur se produit lorsque l'instance à supprimer n'existe pas dans le cache local.
- INVALID_INSTANCE

Cette erreur se produit lorsqu'une clé primaire de l'instance courante n'est pas valide.

- LENGTH_ERROR

  Cette erreur se produit quand le contenu d'une rubrique de l'instance courante a une taille supérieure à la taille autorisée pour la rubrique.

- PARENT_INSTANCE_MISSING

  Cette erreur se produit en cas de sélection d'une instance de noeud dépendant alors que l'instance supérieure est inexistante.

- REFERING_INSTANCE_MISSING

  Cette erreur se produit quand la méthode **`transferReference`** ne trouve pas d'instance dans le **`detail`** du noeud référençant.

- SERVER_UPDATE_REQUIRED

  Cette erreur se produit lorsqu'une méthode est appliquée à une instance alors que l'instance supérieure créée localement n'existe pas encore dans la base. Une mise à jour serveur préalable de l'instance supérieure est requise.

- SUBSCHEMA_ERROR

  Cette erreur se produit lorsqu'une rubrique de l'instance courante est modifiée alors qu'elle n'a pas été renseignée pour cette instance à l'issue d'un accès serveur paramétré avec un sous-schéma.

- UNKNOWN_ASYNCHRONOUS_REQUEST_ID

  Cette erreur se produit lorsqu'une demande de récupération de réponse a été émise et que l'identifiant de réponse est inconnu ou a dépassé le délai d'expiration.

- UNKNOWN_INSTANCE

  Cette erreur se produit en cas de sélection d'une instance inconnue du cache local.

- VALUE_ERROR

  Cette erreur se produit quand le contenu d'une rubrique de l'instance courante n'est pas valide.

- VALUE_REQUIRED

  Cette erreur se produit quand une rubrique de l'instance courante est considérée comme non présente alors qu'elle est obligatoire.

7.2.2. Erreurs serveur

Ces messages d'erreur sont générés en standard par les Services Applicatifs. Ils se répartissent en trois types :

- Erreurs d'accès irrécupérables : **`Erreur d'accès données`**

Il s'agit des erreurs d'accès irrécupérables à un fichier ou une base de données relationnelle.

- Erreurs logiques d'accès : **`Création à tort`** et **`Modif/Annul à tort`**.

- Erreurs de contrôle des champs de la Vue Logique : **`Erreur de valeur`** et **`Rubrique obligatoire`**.

7.2.3. Erreurs système

Les erreurs système sont les suivantes :

- **`Service inconnu`**

Ce message apparaît lorsque le service demandé par la Proxy n'est pas reconnu par le Composant Applicatif.

- **`Longueur de vue erronée`**

Ce message apparaît lorsqu'il y a un changement de format dans une Vue Logique associée à la Proxy et que celle-ci n'a pas été regénérée.

Pour résoudre ce problème, vous devez regénérer la Proxy.

- **`Déphasage composants`**

Cette erreur survient lorsqu'il y a un déphasage entre les composants Client et Serveur.

Pour résoudre ce problème, vous devez regénérer la Proxy.

7.2.4. Erreurs de communication

Si un message d'erreur de communication apparaît, vous devez d'abord en informer le responsable de la communication car la cause peut être un encombrement de ligne, une ligne défectueuse, un serveur indisponible...

Trois messages d'erreur de communication sont susceptibles de s'afficher :

- **`Erreur open serveur`**
- **`Erreur appel serveur`**
- **`Erreur close serveur`**

7.2.5. Erreurs système générées par le Moniteur de Communication ou le Gestionnaire de Services

Ces erreurs sont, pour la plupart, des erreurs internes que vous résoudrez en contactant le support VisualAge Pacbase.

7.2.5.1.1. Erreurs générées par le Moniteur de Communication

Col 1-3	Col 4-9	Col 10-13	Col 21	
bib	mon	LSRV	S	Erreur de longueur du message reçu
bib	mon	PNUM	S	Erreur de structure du paramètre "numéro de service"
bib	mon	PCOD	S	Erreur de structure du paramètre "code de service"
bib	mon	PNOD	S	Code du Dossier inconnu du Moniteur de Communication
bib	mon	PCVS	S	Erreur de structure d'une demande de service en provenance du client
bib	mon	PCVF	S	Erreur de structure du message en provenance du client
bib	mon	WF00	S	Erreur d'accès au fichier de travail

Légende bib = code bibliothèque mon = code Moniteur de Communication S = erreur système

7.2.5.2. Erreurs générées par le Gestionnaire de Services

Col 1-3	Col 4-9	Col 10-13	Col 21	
bib	ges	SRV1	S	Service non trouvé dans le fichier de travail
bib	ges	LNG1	S	Erreur de conversion de la longueur du service sur une requête multi-messages
bib	ges	LNG2	S	Erreur de conversion de la longueur du service sur une requête mono-message
bib	ges	NOS1	S	Erreur de structure du paramètre "numéro de service"
bib	ges	NOS2	S	Erreur de conversion du paramètre "numéro de service"
bib	ges	SRV1	S	Erreur de structure du paramètre "code service"
bib	ges	SRV2	S	Code service inconnu dans le Dossier
bib	ges	DOS1	S	Paramètre "nom du Dossier" absent
bib	ges	DOS2	S	Erreur de longueur du paramètre "nom du Dossier"
bib	ges	VER2	S	Erreur de longueur du pramètre "numéro de version"
bib	ges	NOD1	S	Paramètre "nom du noeud" absent
bib	ges	NOD2	S	Erreur de longueur du paramètre "nom du noeud"
bib	ges	NOD3	S	Nom du noeud inconnu dans le Dossier
bib	ges	TYNO	S	Service non autorisé sur le noeud
bib	ges	SCH1	S	Erreur de structure du paramètre "code sous-schéma"
bib	ges	SCH2	S	Code sous-schéma inconnu dans le Dossier
bib	ges	NOCP	S	Erreur de structure du paramètre "nombre d'occurrences"
bib	ges	NOC1	S	Erreur de longueur sur le paramètre "nombre d'occurrences"
bib	ges	NOC2	S	Erreur de conversion du paramètre "nombre d'occurrences"
bib	ges	EXT1	S	Erreur de structure du paramètre "méthode d'extraction"
bib	ges	EXT2	S	Méthode d'extraction inconnue dans le Dossier
bib	ges	USR1	S	Paramètre "service utilisateur" absent
bib	ges	USR2	S	Erreur de longueur sur le paramètre"service utilisateur"
bib	ges	USR3	S	Service Utilisateur inconnu dans le Dossier
bib	ges	CHK1	S	Paramètre "Check Option" absent
bib	ges	CHK2	S	Erreur de longueur sur le paramètre "Check Option"
bib	ges	RFH1	S	Paramètre "Refresh Option" absent
bib	ges	RFH2	S	Erreur de longueur sur le paramètre"Refresh Option"
bib	ges	LCK1	S	Paramètre "Lock Timestamp" absent
bib	ges	LCK2	S	Erreur de longueur sur le paramètre "Lock Timestamp"

Col 1-3	Col 4-9	Col 10-13	Col 21	
bib	ges	PCV1	S	Erreur de structure du paramètre "Selection Criteria"
bib	ges	PC01	S	Erreur de conversion du paramètre "Selection Criteria"
bib	ges	PILO	S	Erreur d'accès à l'enregistrement pilote du fichier de travail
bib	ges	BUF1	S	Erreur de structure du buffer utilisateur
bib	ges	PC02	S	Erreur de conversion d'un champ du buffer utilisateur
bib	ges	FRWR	S	Erreur d'accès en écriture au fichier de travail
bib	ges	FRW2	S	Erreur d'écriture du dernier enregistrement du fichier de travail
bib	ges	FRRE	S	Erreur d'accès en lecture au fichier de travail avant mise à jour
bib	ges	FRRD	S	Erreur d'accès en lecture au fichier de travail
bib	ges	FRRW	S	Erreur d'accès en mise à jour au fichier de travail
bib	ges	CP01	S	Erreur de structure d'un champ de "Selection Criteria" en provenance du Composant Applicatif
bib	ges	CP02	S	Erreur de structure d'un champ d'une instance de Vue Logique en provenance du Composant Applicatif
bib	ges	ERKY	S	Erreur de structure de la clé du "Selt Message" en provenance du Composant Applicatif
bib	ges	ERLA	S	Erreur de structure du label du "Selt Message" en provenance du Composant Applicatif
bib	ges	PCV!	S	Erreur de structure d'un champ du buffer utilisateur en provenance du Composant Applicatif
bib	ges	PCV3	S	Erreur de structure d'un champ d'une instance de Vue Logique en provenance du client
bib	ges	PC03	S	Erreur de conversion d'un champ d'une instance de Vue Logique en provenance du client
bib	ges	PCV4	S	Erreur de structure d'un champ de "Selection Criteria" en provenance du client
bib	ges	PC05	S	Erreur de conversion d'un champ de "Selection Criteria" en provenance du client
bib	ges	NUVE	S	Erreur du paramètre "numéro de version" dans le Composant Applicatif élémentaire
bib	ges	STRU	S	Erreur du paramètre "structure" dans le Composant Applicatif élémentaire
bib	ges	VIEW	S	Erreur du paramètre "code de la Vue Logique" dans le Composant Applicatif élémentaire
bib	ges	SERV	S	Erreur du paramètre "code opération" dans le Composant Applicatif élémentaire
bib	ges	LTH	S	Erreur du paramètre "longueur" dans le Composant Applicatif élémentaire
bib	ges	PCV5	S	Erreur de structure du code action d'une instance de Vue Logique à mettre à jour
bib	ges	PCV6	S	Erreur de structure d'un champ d'une instance de Vue Logique à mettre à jour en provenance du client
bib	ges	PCV7	S	Erreur de structure du vecteur de présence d'un champ d'une instance de Vue Logique à mettre à jour
bib	ges	PC06	S	Erreur de conversion d'un champ d'une instance de Vue Logique à mettre à jour en provenance d'un Composant Applicatif élémentaire
bib	ges	ERK1	S	Erreur de structure de la clé de message d'erreur utilisateur
bib	ges	ERL1	S	Erreur de structure du label de message d'erreur utilisateur
bib	ges	OCNB	S	Erreur de structure, dans le message d'erreur utilisateur, du code champ sur lequel porte l'erreur
bib	ges	DANA	S	Erreur de structure du code de champ erroné dans le message d'erreur utilisateur
bib	ges	PCVS	S	Erreur de structure d'une demande de service en provenance du client
bib	ges	PCVF	S	Erreur de structure du message en provenance du client
bib	ges	WF00	S	Erreur d'accès au fichier de travail

Légende bib = code bibliothèque ges = code Gestionnaire de services S = erreur système

7.3. Gestion de la communication

7.3.1. Composants du middleware

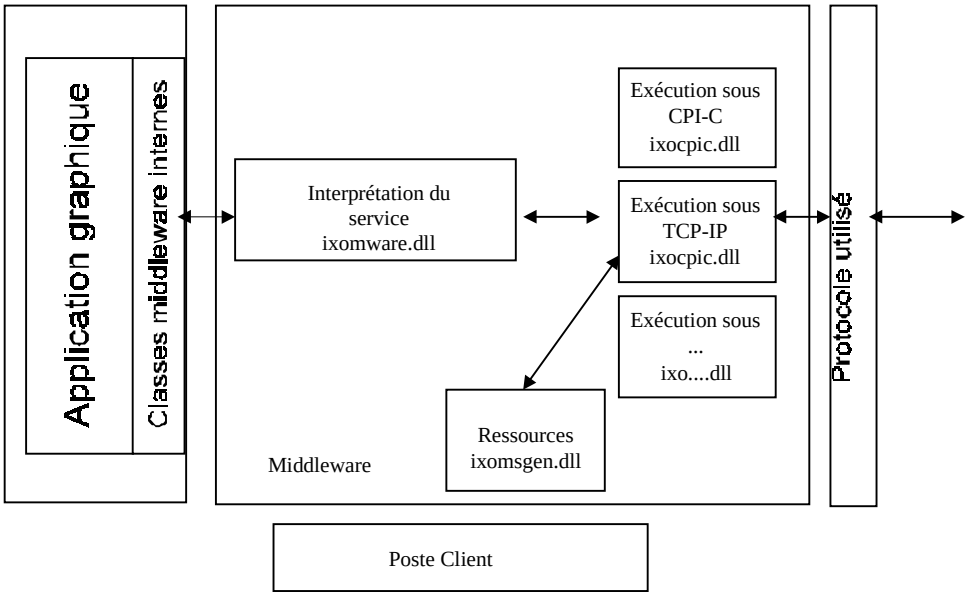
Le middleware de Pacbench Client/Serveur correspond à une application C et C++ dont les fonctions sont réparties dans trois types de DLLs :

- le premier type regroupe les fonctions d'interfaçage avec les langages C et C++ et les fonctions d'interprétation des services du middleware et d'aiguillage vers des librairies de fonctions spécialisées pour un protocole de communication spécifique.
- le deuxième type regroupe les fonctions d'exécution des services du middleware de Pacbench Client/Serveur pour un protocole de communication spécifique.
- le troisième type regroupe des fonctions de gestion de ressources telles que la traduction en clair des exceptions rencontrées.

Nom de la DLL	Fonction	Plate-forme	Protocole
ixomware.dll	Type-1	Windows95/NT	Tous
ixocics.dll	Type-2	Windows95/NT	CICS-ECI
ixocpic.dll	Type-2	Windows95/NT	CPI-C
ixotux.dll	Type-2	Windows95/NT	TUXEDO
ixosock.dll	Type-2	Windows95/NT	SOCKET TCP-IP
ixoloc.dll	Type-2	Windows95/NT	Appel de DLL COBOL
ixomqs.dll	Type-2	Windows95/NT	MQSERIES
ixomsgen.dll	Type-3	Windows95/NT	Tous
csoom30.dll	runtime C++	Windows95/NT	Tous
msvcrt20.dll	runtime C++	Windows95/NT	Tous

7.3.2. Schéma de traitement d’une requête

Dans votre client, les services du middleware sont exécutés à partir d'un ensemble de classes de communication spécifiques livrées à l'installation du middleware.



7.3.3. Définition du contexte d'utilisation

La gestion de la communication nécessite la définition du contexte d'utilisation du middleware. Ce contexte est défini dans un fichier spécifique dans lequel sont indiqués, pour chaque Dossier généré, une ou plusieurs localisations.

Une localisation, ou *location* permet de spécifier les éléments qui autorisent les échanges de données entre les applications clientes et les gestionnaires de services et ce, que ce soit dans une étape de développement de l’application cliente ou lors de son exploitation. Par exemple, une application cliente peut s’adresser à un serveur sans trace pour une version d’exploitation.

Pour un Dossier, une localisation est composée :

- de l’identifiant de la localisation. Cet identifiant est sensible à la distinction majuscules/minuscules.
- du nom externe du moniteur de communication.
- de la longueur physique maximum d’un message d’échange.
- d’un nombre variable de paramètres qui permettent de faire fonctionner le middleware (adresse IP, trace...).

Les localisations sont spécifiées dans le fichier **`VAPLOCAT.INI`** (ce nom de fichier en majuscules est réservé).

7.3.3.1. Structure du fichier VAPLOCAT.INI

Le fichier des localisations est structuré en sections et sous-sections.

Une section correspond à un Dossier et est identifiée par le code de celui-ci, placé entre [...].

Une sous-section correspond à une localisation et est identifiée par le nom de l'environnement spécifié pour l'option **LOCATION** dans la fenêtre **Commentaires** du Dossier. L'identifiant d'une sous-section est placé entre <...>.

Dans une sous-section, les différents paramètres sont représentés par des mots-clés.

Les mots-clés utilisés sont les suivants :

Mot-clé	Signification
COMMENT	Commentaire
LENGTH	Longueur physique du message (générée)
MONITOR	Nom externe du moniteur de communication (généré)
MWADDRESS	Adresse du serveur (pour SOCKET, TCPMVS, TCIS, TCPIMSI et TCPIMSE) Obligatoire pour ces protocoles Les formats suivants sont possibles: - Codification décimale : * adresse IP ou alias de l'adresse IP * blanc * numéro de port en décimal - Codification hexadécimale: * [1, 2]: '0x'' * [3, 6]: domaine AF_INET * [7,10]: numéro de port en hexadécimal * [11, 18]: adresse IP
MWARE	Protocole de communication **Valeurs** : **Protocoles** TUXEDO TUXEDO XA ou non XA CICS CICS ECI extend ou nonextend LOCAL Serveur sur la station Client CPIC CICS CPI-C/APPC ou IMS CPI-C/APPC SOCKET TCP-IP socket pour UNIX ou WINDOWS NT TCIS TCIS MQSERIES MQSERIES TCPMVS TCP-IP SOCKET pour MVS/CICS
MWBUFFERTYPE	Type de buffer (pour TUXEDO) Valeurs : - CARRAY (défaut) - FML
MWCODEPAGE	Code page du serveur - Facultatif
MWFMLNAME	Nom de fichier FML (pour TUXEDO). Attention : vous devez aussi indiquer le chemin dans la variable d'environnement FLDTBLDIR et le nom du fichier dans la variable d'environnement FIELDTBLS.
MWMAXREPLY	Nombre maximum de requêtes asynchrones en attente de réponse - Facultatif - Numérique
MWQUEUEMANAGER	Nom du manager de queues pour protocole asynchrone (MQSERIES) - Obligatoire pour ce protocole - Alphanumérique - 48 caractères max.
MWREPLYQUEUE	Nom de la queue de message en réception pour protocole asynchrone (MQSERIES) - Obligatoire pour ce protocole - Alphanumérique - 48 caractères max.

Mot-clé	Signification
MWREPORTQUEUE	Nom de la queue de message en réponse de mise à jour pour protocole asynchrone (MQSERIES) - Obligatoire pour ce protocole - Alphanumérique - 48 caractères max.
MWREQUESTEXPIRY	Durée de vie d'une requête asynchrone - Facultatif - Numérique
MWREQUESTQUEUE	Nom de la queue de message en émission pour protocole asynchrone (MQSERIES) - Obligatoire pour ce protocole - Alphanumérique - 48 caractères max.
MWTIMEOUT	Time out géré par le middleware (en secondes) - Facultatif - Numérique - Valeur: [0, 32767]
MWTRANSID	Code de transaction CICS sur 4 caractères : - obligatoire pour le protocole TCP-IP socket MVS - facultatif pour le protocole CICS/ECI • Si le paramètre MWTRANSID est absent ou non valorisé, la transaction CPMI est automatiquement exécutée sur CICS. Dans ce cas, le paramètre MONITOR est obligatoire et doit correspondre à un programme exécutable sous CICS. • Si le paramètre est valorisé, le code de transaction sera exécuté sous CICS pour chaque requête utilisant cette location. Dans ce cas, la valeur du paramètre MONITOR n'est pas exploitée.

Chaque ligne de fichier ne doit contenir qu'un paramètre.

Exemple de fichier de localisation :

```
[FOCLNT]
<Location1>
COMMENT=moniteur d'exploitation
MONITOR=CLCOMM
LENGTH=8192
MWCODEPAGE=850
MWARE=SOCKET
MWADDRESS=0x00021770C0060A5D
<Location2>
COMMENT=moniteur avec trace
MONITOR=CLCOM2
LENGTH=8192
MWARE=SOCKET
MWADDRESS=0x00021770C0060A5D
<Location3>
COMMENT=moniteur local
MONITOR=CLCOMM
LENGTH=8192
MWARE=LOCAL
```

7.3.3.2. Mise en oeuvre

Le fichier **VAPLOCAT.INI** est obligatoire en phase de développement et en phase d'exploitation.

En cas d'absence de ce fichier ou si la syntaxe en est incorrecte, le Gestionnaire d'échanges ne peut être instancié. Pour connaître le paramètre erroné, consultez le fichier trace.

En phase de développement, le générateur de Proxy Vues de Dossier assure la création ou la mise à jour du fichier **VAPLOCAT.INI.**

- S'il n'existe pas, le générateur le crée dans le répertoire courant de VisualAge.

  Pour chaque localisation, les valeurs des paramètres **MONITOR** et **LENGTH** (générés) sont automatiquement positionnées.

  Le développeur a la charge de positionner les autres paramètres.
- S'il existe déjà (dans le répertoire de VisualAge), il est automatiquement chargé. Pour chaque Dossier, le générateur procède alors à une comparaison entre les localisations du fichier d'extraction et les localisations existantes. Il assure la mise à jour du fichier existant :
  - en ajoutant automatiquement les nouvelles localisations,
  - en supprimant automatiquement les localisations qui n'existent plus dans le fichier d'extraction,
  - en modifiant les localisations existantes par la prise en compte des modifications contenues dans le nouveau fichier d'extraction.
    En cas de modification, les paramètres positionnés par le développeur restent inchangés.

7.4. Déploiement de l'application

Pour déployer l'application, suivez les explications indiquées dans la documentation de votre outil de développement graphique.

Mais vous devez aussi installer certains fichiers liés à l'utilisation de Pacbench C/S.

- **Pour une application Web**

Le poste de l'utilisateur final doit être simplement équipé d'un navigateur.

- **Pour une application *standalone*** :
  - Si aucune gateway n'est utilisée, vous devez installer, sur le poste de l'utilisateur final :
    - les DLL du middleware,
    - **VAPLOCAT.INI**,
    - **CHARCONV.TXT** (conversion du code page),
    - **VAPRUNTIME.DLL**,
    - **MWADAPTER.DLL**.
  - Si une gateway est utilisée, vous devez installer, sur le poste où est installée la gateway :
    - **GATEWAY.EXE**,
    - les DLL du middleware,
    - **VAPLOCAT.INI**,
    - **CHARCONV.TXT** (conversion du code page),
    - **GWADAPTER.DLL**.

    Dans ce cas, vous ne devez installer aucun de ces fichiers sur le poste de l'utilisateur final. En revanche, vous devez y installer le fichier **VAPRUNTIME.DLL**.

8. Pont Pacbench Client / Serveur

8.1. Introduction

Le Pont de Pacbench Client/Serveur vous permet de :

- disposer d’une base de référence unique, le Référentiel.
- stocker dans cette base de référence les objets provenant de VisualAge qui bénéficient ainsi des avantages des fonctions de gestion, de références croisées et de sécurité du Référentiel.

Le Pont est constitué d'une procédure de Sauvegarde, d'une procédure d'Epuration et d'une procédure de Restauration (pour VisualAge Smalltalk uniquement).

- La procédure de Sauvegarde permet de stocker dans le Référentiel les références croisées des objets provenant de VisualAge selon un modèle d'information décrit ci-après.

  Pour Smalltalk, elle permet aussi de sauvegarder le source des objets.

  Les entités VisualAge dédiées au stockage sont accessibles en consultation uniquement, à partir de la **`Station de Travail VisualAge Pacbase`** (Module **`Pacbench`**).

- La procédure d'Epuration consiste à rechercher les entités VisualAge non utilisées et à les supprimer de la base.

- La procédure de Restauration Smalltalk permet de restituer dans l’environnement VisualAge les objets stockés dans le Référentiel.

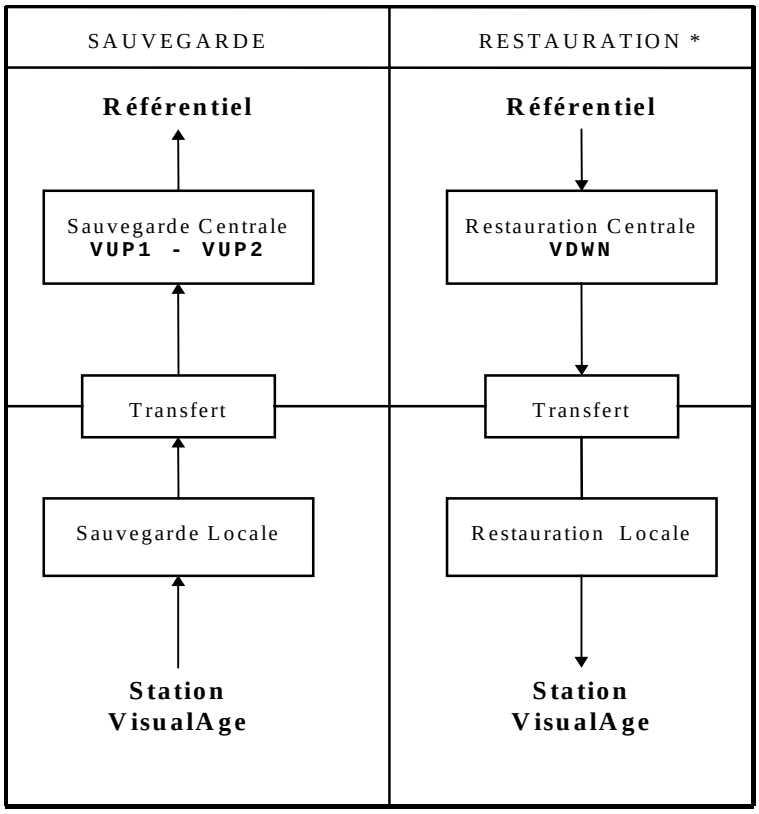
Les procédures de Sauvegarde et de Restauration comprennent deux parties :

- une partie s’exécute en local dans l’environnement VisualAge,
- l'autre partie s'exécute en site central par des procédures batch.

La procédure d'Epuration est une procédure batch.

Pour l'exécution de ces procédures batch, une autorisation d'accès de niveau 4 est requise. Les spécificités techniques relevant de la plate-forme sont documentées dans le *Manuel d’Exploitation* correspondant.

Les étapes de la Sauvegarde et de la Restauration



* La procédure de restauration est disponible uniquement pour la station VisualAge Smalltalk .

8.2. Modèle d'information de Pacbench Client/Serveur (Client graphique)

L'ouverture du Pont de Pacbench Client/Serveur requiert l'installation dans le Référentiel d'Entités Utilisateur dédiées au stockage des objets VisualAge.

Pour les détails techniques relatifs à cette installation, référez-vous au Manuel d'exploitation *VisualAge Pacbase - Procédures Batch - Guide de l'Administrateur*, chapitre *Intégration du Dictionnaire VisualAge*.

Le Modèle d'Information de Pacbench Client/Serveur avec un composant client graphique comprend deux sous-schémas :

- le sous-schéma Services Applicatifs,
- le sous-schéma Interface Utilisateur (mode graphique).

Ces deux sous-schémas contiennent des entités du Référentiel et des entités spécifiques à l'environnement de développement de clients graphiques, séparées par un arc de cercle en pointillé. Ces sous-schémas comportent également les chaînages entre ces différentes entités, leurs relations et les cardinalités de ces dernières.

8.2.1. Sous-schéma Services Applicatifs

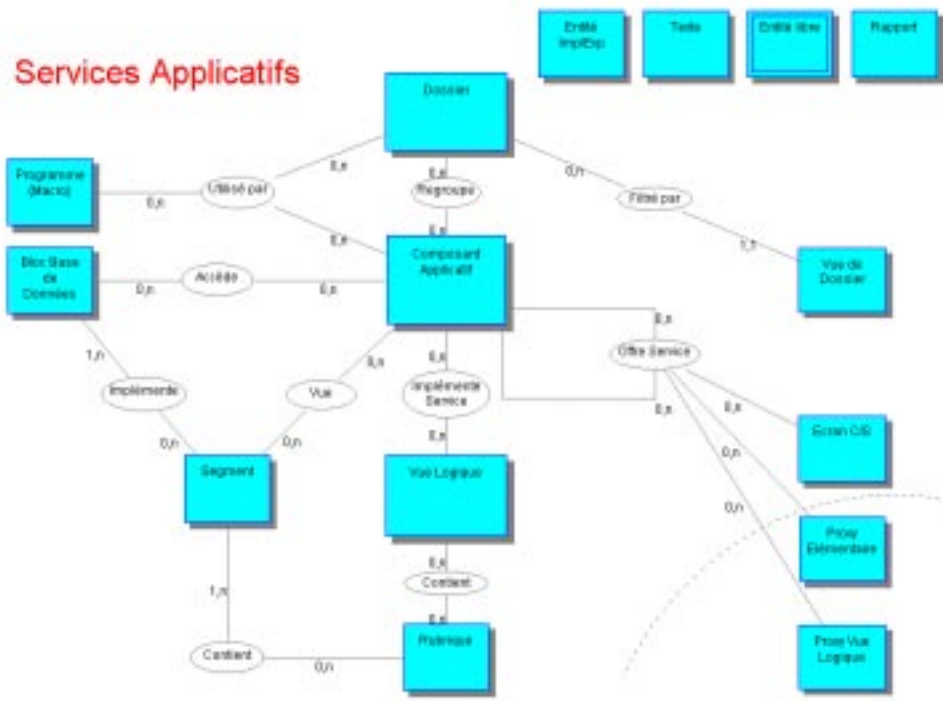
Ce sous-schéma est accessible à partir de la fenêtre **Gestionnaire de la Station de Travail** (Module **Pacbench**, menu **Métamodèle**, choix **Services Applicatifs**).

Du côté serveur, il comporte les entités Dossier, Composant Applicatif, Vue Logique, Rubrique, Segment, Bloc Base de Données et Programme.

Du côté client, il comporte :

- pour les clients mode caractère, les entités Vue de Dossier et Ecran C/S,
- pour les clients mode graphique, les entités Proxy Vue Logique (mode simple) et Proxy Elémentaire (mode dossier).

Les clients mode caractère ne sont pas documentés dans ce manuel. Pour des informations, reportez-vous au *Guide Utilisateur Pacbench C/S, Vol. II – Services Applicatifs*.



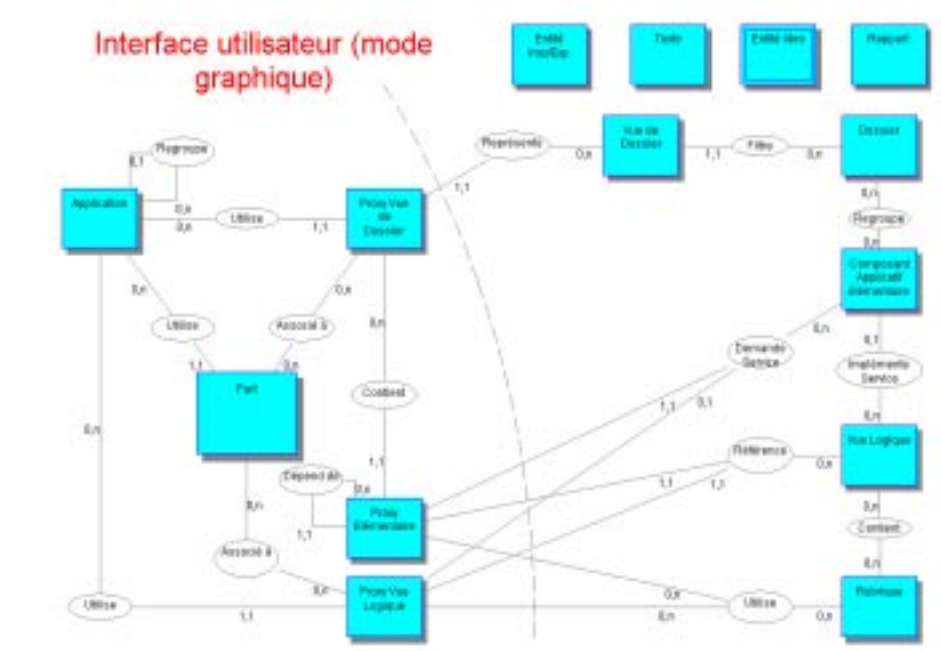
8.2.2. Sous-schéma Interface Utilisateur (mode graphique)

Ce sous-schéma est accessible à partir de la fenêtre `Gestionnaire de la Station de Travail` via le Module `Pacbench`, menu `Métamodèle`, choix `Interface Utilisateur (mode graphique)`.

Du côté serveur, il comporte les entités Dossier, Vue de Dossier, Composant Applicatif élémentaire (gère une seule Vue Logique), Vue Logique et Rubrique.

Du coté client, il comporte :

- les entités Part et Application, communes aux modes dossier et simple,
- les entités Proxy Vue de Dossier et Proxy Elémentaire, spécifiques au mode dossier,
- l'entité Proxy Vue Logique, spécifique au mode simple.



Pour plus de détails sur les modèles d'information et les conventions qui y sont utilisées, reportez-vous au Manuel de Référence *Station de Travail VisualAge Pacbase*, chapitre *Les Métamodèles*.

8.2.3. Entités VisualAge

Les entités Part, Application, Proxy Vue de Dossier, Proxy Elémentaire et Proxy Vue Logique sont des Entités Utilisateur dédiées au stockage des objets VisualAge correspondants.

Pour chacune de ces entités, vous trouverez dans le paragraphe qui lui est consacré :

- la Définition,
- le contenu de leurs différentes Descriptions,
- les chaînages avec les autres Entités Utilisateur VisualAge,
- et le code d'appel dans le Référentiel.

Ces entités sont accessibles au travers de la **Station de Travail VisualAge Pacbase (Module Pacbench)** à partir de la boîte **Entité** de la fenêtre **Gestionnaire de la Station de Travail VisualAge Pacbase**. Elles sont disponibles en consultation; vous ne devez en aucun cas les modifier.

8.2.3.1. Entité Application

L'entité Application est un regroupement de Parts qui décrivent une application VisualAge.

Définition

Code de l'occurrence

Libellé : Identifiant VisualAge

Mots clés

Code de l'Application parente

Libellé de l'Application parente

Descriptions

-D1 à -D6	Code source
-D9	Identifiant VisualAge

Chaînages

Applications filles

Parts de l'application

Proxy Vues de Dossier utilisées

Proxy Vues Logiques utilisées

Code d'appel

$77

8.2.3.2. Entité Part

L’entité Part permet de stocker le source d'un part VisualAge.

Définition

Code de l’occurrence

Libellé : Identifiant VisualAge

Mots clés

Code de l'application associée

Descriptions

-D1 à -D6	Code source
-D8	Liste des Proxy Vues de Dossier associées
-D7	Liste des Proxy Vues Logiques associées
-D9	Identifiant VisualAge

Chaînages

Application

Code d’appel

```
$78
```

8.2.3.3. Entité Proxy Vue de Dossier

L'entité Proxy Vue de Dossier permet d’établir le lien avec la Vue de Dossier et les Composants Applicatifs élémentaires à partir desquels elle a été générée.

Définition

Code de l’occurrence

Libellé : Identifiant VisualAge

Mots clés

Code de la Vue de Dossier associée

Libellé de la Vue de Dossier associée

Code de l'Application associée

Libellé de l'Application associée

Descriptions

-D8	Proxy Elémentaire
-D9	Identifiant Proxy Vue de Dossier VisualAge

Chaînages

Part

Code d’appel

$7X

8.2.3.4. Entité Proxy Elémentaire

L'entité Proxy Elémentaire n'a pas d'existence autonome et fait partie d'une Proxy Vue de Dossier. Elle permet d'établir le lien avec la Vue Logique et le Composant Applicatif élémentaire à partir desquels elle a été générée.

Définition

Code de l'occurrence

Libellé : Identifiant VisualAge

Mots clés

Code de la Vue Logique référencée

Libellé de la Vue Logique référencée

Code du Composant Applicatif élémentaire appelé

Libellé du Composant Applicatif élémentaire appelé

Code de la Proxy Elémentaire parente

Type de dépendance (R/L/D)

Libellé de la Proxy Elémentaire parente

Descriptions

-D5	Liste des Rubriques appelées
-D9	Identifiant Proxy Elémentaire VisualAge

Chaînages

Proxy Vue de Dossier

Proxy Dépendantes

Code d'appel

$7Y

8.2.3.5. Entité Proxy Vue Logique

L'entité Proxy Vue Logique permet d’établir le lien avec la Vue Logique et le Composant Applicatif à partir desquels elle a été générée.

Définition

Code de l’occurrence

Libellé : Identifiant VisualAge

Mots clés

Code de la Vue Logique référencée

Libellé de la Vue Logique référencée

Code du Composant Applicatif associé

Libellé du Composant Applicatif associé

Code de l'Application associée

Libellé de l'Application associée

Descriptions

-D5	Liste des Rubriques appelées
-D9	Identifiant Proxy Vue Logique VisualAge

Chaînages

Part

Code d’appel

$79

8.3. Principes de fonctionnement du Pont

Une base VisualAge Pacbase est constituée d'un réseau de bibliothèques de données hiérarchiquement organisées et formant une structure arborescente.

Les sessions historisées permettent de gérer en parallèle plusieurs images différentes de ce même réseau. Ces images peuvent être évolutives (session historisée de type 'T' ou session courante) ou figées (session historisée de type 'H').

Les données manipulées dans VisualAge, qu'elles soient générées à partir du Référentiel ou créées directement dans VisualAge, sont stockées dans le Référentiel sous forme d'occurrences d'Entités Utilisateur créées lors de la procédure de Sauvegarde.

Ces occurrences peuvent être consultées mais ne peuvent en aucun cas être modifiées.

8.4. Sauvegarde : Remontée des données dans le Référentiel

La procédure de Sauvegarde permet de stocker dans le Référentiel les composants générés, après leur manipulation dans VisualAge, et ceux créés en local dans l'environnement VisualAge.

La procédure de Sauvegarde se compose de deux étapes, l'une s'exécutant en local, l'autre sur le serveur.

8.4.1. Sauvegarde locale Java

8.4.1.1. Fonctionnalités

Le pont VisualAge Pacbase for Java fonctionne sur toutes les plate-formes supportant la version 1.1 de Java Development Kit (JDK).

Il permet de créer un fichier qui contient les références croisées entre les serveurs et les applications Java. Ce fichier peut être importé dans le Référentiel.

8.4.1.2. Principe du CLASSPATH

La variable CLASSPATH est utilisée par le pont VisualAge Pacbase for Java afin d'analyser les classes installées sur le poste client. La valeur initiale de cette variable est lue à partir de l'environnement machine mais vous pouvez modifier cette valeur.

Lors de la première analyse, le programme recherche directement dans le chemin d'accès tous les fichiers dont l'extension est `.class` ainsi que les fichiers `.jar` et `.zip`. Toutes les classes qui ne référencent pas les objets Proxy VisualAge Pacbase sont immédiatement écartées. Cela comprend les références aux packages suivants :

- Packages Java,
- Tous les packages dont le nom commence par **sun**,

- Tous les packages contenant les classes VisualAge Pacbase,

Ensuite tous les fichiers retenus sont analysés. Lors de cette opération, chaque fichier est décompilé afin de retrouver les informations suivantes :

- Le nom de la classe,
- La classe mère,
- Les interfaces implémentées par la classe,
- Toutes les autres classes référencées par la classe (cela inclus les liens entre les classes ainsi que les classes référencées comme argument de méthode, type retour, type champs et classe ancêtre).
- les constantes VisualAge Pacbase contenues dans les champs **`cvapFolder, cvapServe`** et **`cvapView`**.

Ces informations sont enregistrées dans une table qui sera utilisée pour la sauvegarde des références croisées.

Lorsque toutes les classes ont été analysées, le programme affiche une liste des packages trouvés. Le programme affiche les Proxy Vues de Dossier dans une liste Dossiers.

Le fichier résultat de la sauvegarde contiendra une liste de toutes les Proxy Vues de Dossier que vous aurez sélectionnées et de toutes les classes qui les référencent directement ou indirectement. Dans le cas des références indirectes, la Proxy n'est pas référencée directement dans la classe mais une classe référence une autre classe qui elle-même référence la Proxy.

8.4.1.3. Lancement du programme

Pour lancer le pont VisualAge Pacbase for Java, exécutez la commande suivante dans une session DOS :

`<Java VM executable>` **`com.ibm.vap.bridge.Main [options]`**

Option **`-lang`** ***`<language> :`***

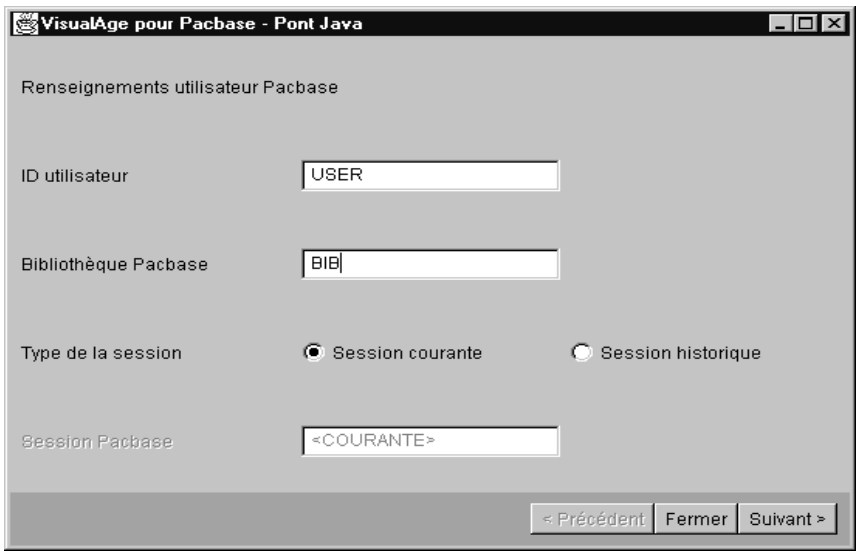
L'option langue permet de choisir une langue pour la présentation de l'interface utilisateur de l'application : **`Fr`** pour français et **`Eng`** pour anglais. Par défaut la langue du système est utilisée.

Si le pont a été préalablement importé dans la Station VisualAge, vous pouvez le lancer de la façon suivante :

- Dans le Workbench, sélectionnez le Projet dans lequel le pont a été importé.
- Affichez son menu contextuel, sélectionnez le sous-menu **`Run`**, puis le choix **`Run Main`**….
- La fenêtre **`Command Line Argument`** s'affiche avec l'option langue par défaut : **`-lang fr`**.
- Cliquez sur **`Run`**.

8.4.1.4. Identification de l'utilisateur

Au lancement du programme, la fenêtre suivante s'ouvre :



Vous devez entrer dans cette fenêtre les informations liées au contexte utilisateur :

- **ID utilisateur** : votre identifiant VisualAge Pacbase (8 caractères maximum)
- **Bibliothèque VisualAge Pacbase** : La bibliothèque affectée au Référentiel VisualAge Pacbase (3 caractères maximum)
- **Type de session** : courante ou historique
- Dans la zone **Session VisualAge Pacbase** entrez les quatre chiffres qui identifient une session historisée.

Cliquez sur le bouton **Suivant** pour accéder à la fenêtre qui vous permettra de lancer une recherche des classes.

8.4.1.5. Options de recherche

Après avoir entré les informations relatives au contexte utilisateur, appuyez sur le bouton **Suivant**.

La fenêtre suivante s'ouvre :

Plusieurs possibilités vous sont offertes, vous pouvez :

- Lancer une recherche des classes à partir du chemin spécifié dans la variable classpath.
- Lancer l'analyse à partir d'un fichier contenant une table de classes résultant d'une recherche antérieure. Dans ce cas, vous devez spécifier le format du fichier à analyser : fichier .**SER** ou fichier .**ZIP**.

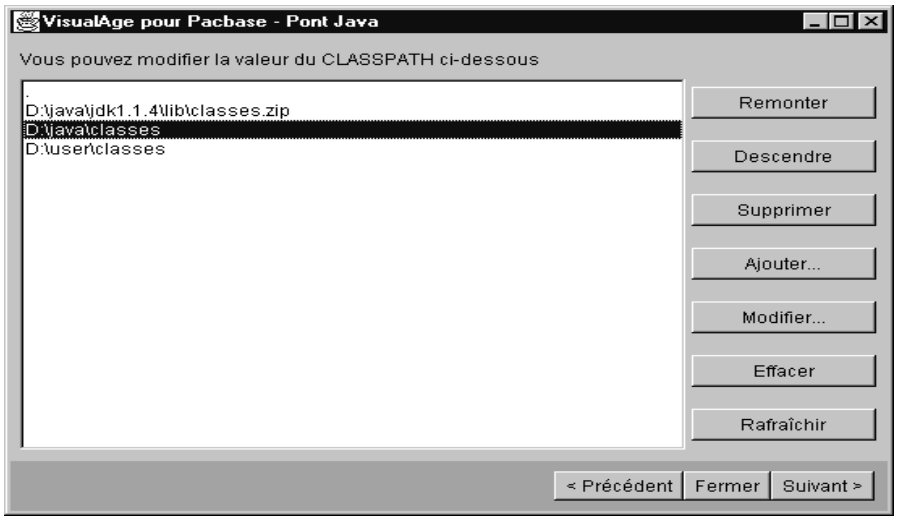
Lorsque vous sélectionnez l'option **Analyse des classes figurant dans le classpath**, et que vous cliquez sur **Suivant**, le programme exécute une recherche globale de toutes les classes disponibles dans la station cliente.

Vous pouvez toutefois demander une recherche ciblée en cochant la case **Modifier le CLASSPATH avant l'analyse**. Lorsque vous cliquez sur le bouton **Suivant**, vous accédez à une fenêtre dans laquelle vous pouvez modifier la valeur de la variable classpath.

Vous avez également la possibilité d'enregistrer le résultat de la recherche dans un fichier afin de l'utiliser ultérieurement. Le programme fournit deux formats d'enregistrement pour ce fichier, compressé ou non-compressé : **zip** ou **SER.**

8.4.1.6. Modification de la valeur du classpath

Si vous avez choisi de modifier le CLASSPATH avant l'analyse, la fenêtre suivante s'ouvre.



Cette fenêtre vous sert à modifier, ajouter, ou supprimer les composants d'un CLASSPATH. Ces changements auront une incidence seulement sur l'analyse des classes mais n'affecteront en aucun cas la variable d'environnement du système ni le comportement du programme.

Les boutons de la fenêtre de l'éditeur de CLASSPATH correspondent aux fonctions suivantes :

- **Remonter** : permet de remonter dans le liste
- **Descendre** : permet de descendre dans la liste.
- **Supprimer** : cette option permet de supprimer un composant de la liste.
- **Ajouter** : affiche une boîte de dialogue qui permet de saisir le nom du nouveau fichier à ajouter à la liste.
- **Modifier** : affiche une boîte de dialogue qui permet de modifier le nom d'un fichier ou d'un répertoire de la liste.
- **Effacer** : permet d'effacer tous les fichiers de la liste.
- **Rafraîchir** : permet de remettre le classpath à sa valeur initiale.

Appuyez sur le bouton **Suivant** pour lancer l'analyse du classpath et accéder à la fenêtre suivante.

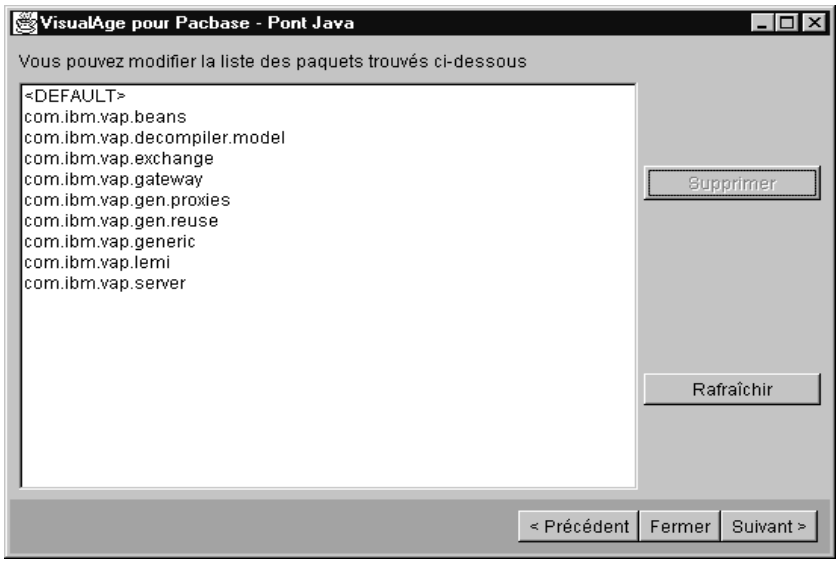
8.4.1.7. Modification de la liste des packages

Lorsque l'analyse des classes est terminée ou la table des classes a été lue, la fenêtre permettant de modifier les packages s'ouvre.

Vous pouvez utiliser cette fenêtre pour supprimer les packages que vous ne souhaitez pas voir apparaître dans le fichier résultat de la sauvegarde.

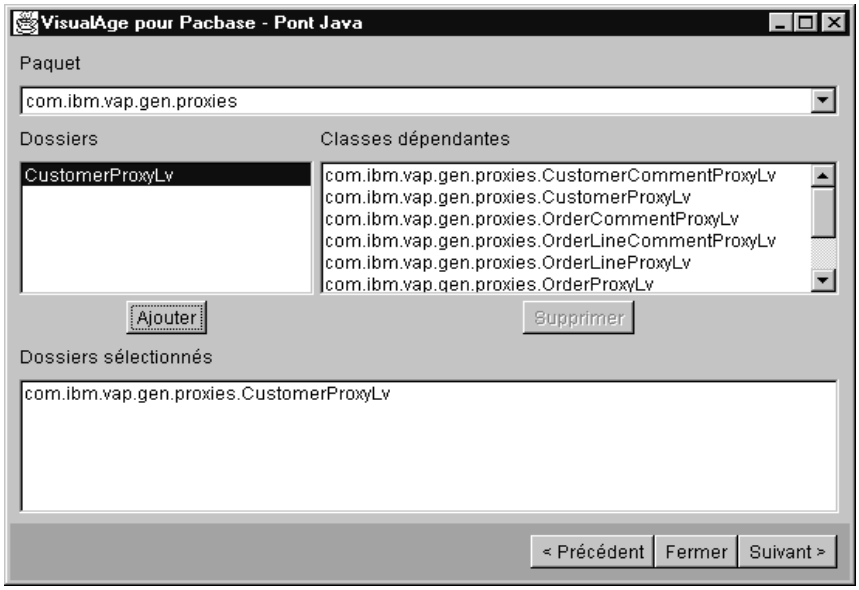
La fonction de cette fenêtre est la même que celle de la fenêtre précédente, elle permet d'exécuter les actions suivantes :

- **Supprimer** : supprimer un package.
- **Rafraîchir** : redonner à la liste de packages sa valeur initiale.



8.4.1.8. Sélection des Proxy Vues de Dossier

Une fois la liste de packages modifiée, le programme affiche une fenêtre dans laquelle s'affiche une liste des packages trouvés.

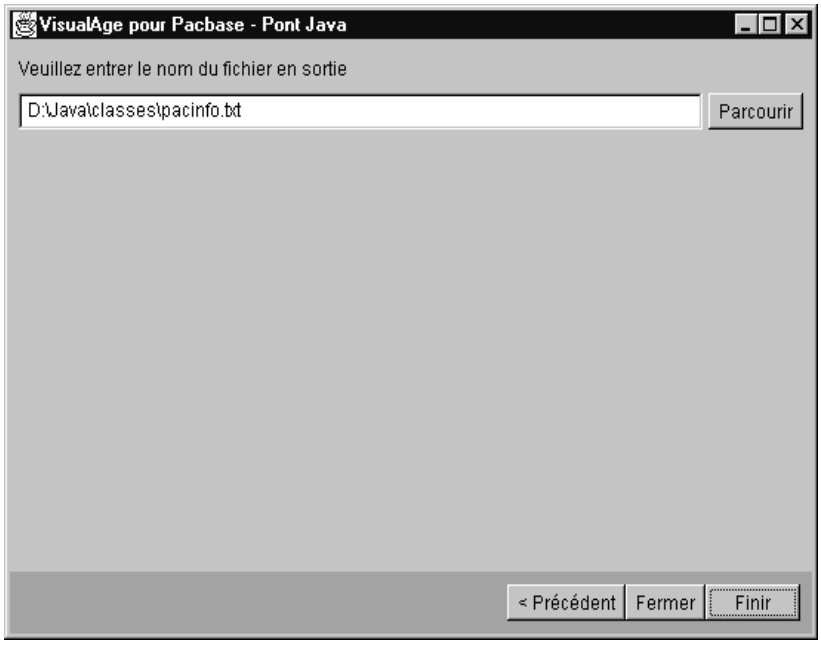


Les packages sont affichés dans une liste déroulante en haut de la fenêtre. Lorsqu'un package est sélectionné, les Proxy Vues de Dossier qu'il contient sont affichées dans la liste **Dossiers**. Vous pouvez alors sélectionner les Proxy de votre choix, les ajouter à la liste ou les supprimer.

Ensuite cliquez sur le bouton **Suivant** pour accéder à la fenêtre permettant la saisie du nom du fichier de sauvegarde.

8.4.1.9. Sélection et génération du fichier de sauvegarde

Cette fenêtre permet de saisir le nom du fichier de sauvegarde. Une fois le fichier sélectionné, cliquez sur le bouton **Finir** afin de générer le fichier.



8.4.2. Sauvegarde locale Smalltalk

8.4.2.1. Fonctionnalités

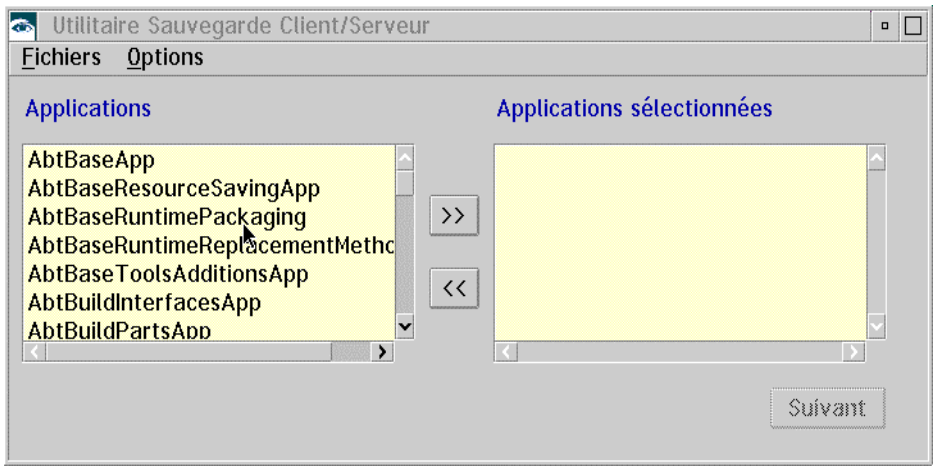
La Sauvegarde locale analyse le contenu du fichier source de l'Application sélectionnée, formate les mouvements de sauvegarde et prépare la création des chaînages entre les entités.

En sortie, elle produit un fichier de sauvegarde de mouvements de mise à jour qui sera utilisé en entrée de la procédure **VUP1**.

Avant de lancer la Sauvegarde locale, afin d'éviter l'apparition de certaines erreurs, vérifiez que vous avez tenu compte des conseils donnés au chapitre **6**.

8.4.2.2. Fenêtre principale de la Sauvegarde Locale

Pour sélectionner l'application à sauvegarder, ouvrez la fenêtre de sauvegarde à partir du menu **Outils** de VisualAge Organizer, choix **Sauvegarde.**



Cette fenêtre fournit la liste de toutes les applications chargées. Les chevrons permettent de transférer les applications d'une liste à l'autre.

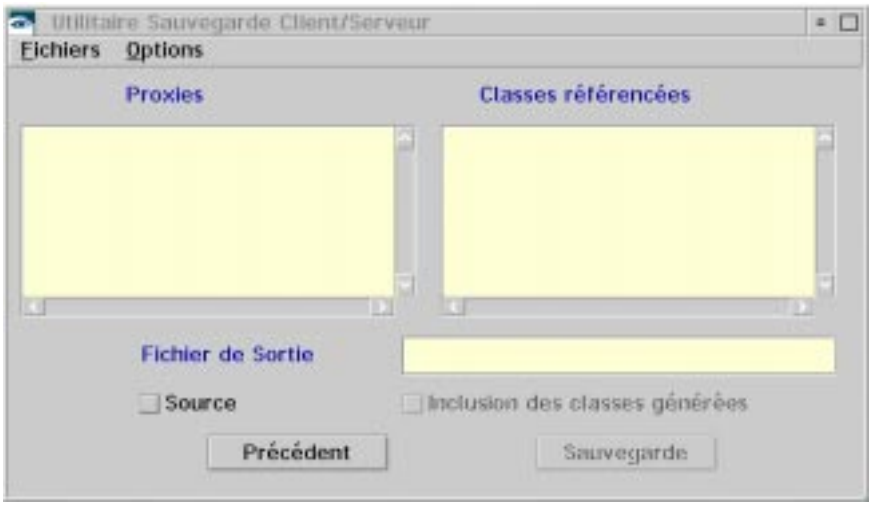
Vous pouvez procéder à un choix de langue à partir du menu **Options**. La langue sélectionnée est celle qui sera utilisée pour la présentation de la fenêtre.

Le menu **Option** offre également le choix **Contexte utilisateur** qui permet d'afficher la fenêtre de saisie des informations contextuelles de VisualAge Pacbase.

Le menu **Fichiers** contient le choix **Quitter** qui permet de sortir du pont.

Le bouton **Next** permet d'afficher la fenêtre dans laquelle apparaîtra la liste des objets Proxy référencés par les applications sélectionnées.

8.4.2.3. Fenêtre listant les objets Proxy



Cette fenêtre affiche la liste des objets Proxy référencés par les applications sélectionnées dans la fenêtre prédédente.

Un double clic sur une Proxy permet d'afficher la liste des classes référencées par la Proxy sélectionnée. Les références sont soit directes soit indirectes, dans ce dernier cas, la Proxy n'est pas référencée directement dans la classe mais une classe référence une autre classe qui elle-même référence la Proxy.

Saisissez dans la zone **Fichier de sortie**, le nom du fichier d'extraction qui contiendra le résultat de la sauvegarde et qui sera utilisé en entrée de la procédure **VUP1**. Si vous souhaitez consulter à nouveau la liste des fichiers, elle est accessible via le menu **Fichiers**.

La fenêtre contient deux options :

- l'option **Source** qui permet de préciser si vous voulez remonter les sources des applications que vous avez sélectionnées dans la liste **Classes référencées**.
- l'option **Inclusion des classes générées** qui permet de spécifier si vous voulez remonter le source des classes générées des objets Proxy présents dans une application dont on remonte le source.

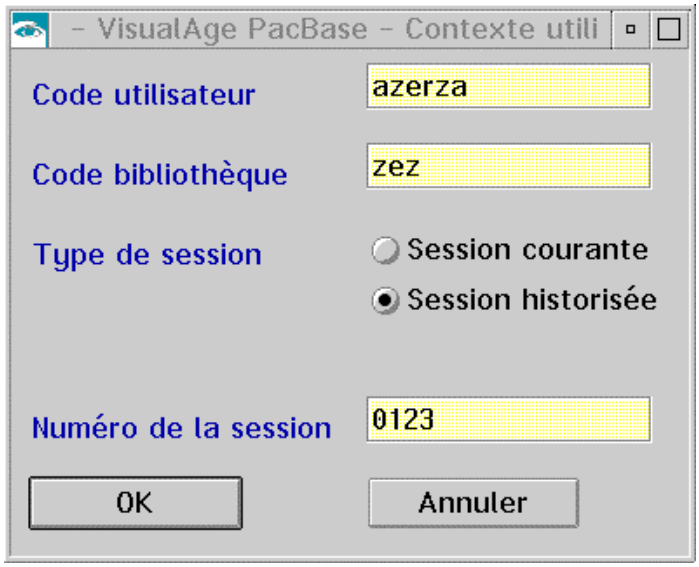
A partir du menu **Fichiers**, vous pouvez choisir la langue de présentation des fenêtres (anglais, français) et accéder à la fenêtre de saisie des informations contextuelles de VisualAge Pacbase par le choix **Contexte utilisateur.**

Pour sauvegarder, appuyez sur le bouton **Sauvegarde.** Vous lancez ainsi la constitution du fichier de sauvegarde.

Le bouton **Précédent** permet de revenir à la fenêtre principale de sauvegarde.

8.4.2.4. Contexte utilisateur

La fenêtre suivante s'ouvre automatiquement après avoir lancé la sauvegarde du fichier. Elle permet de définir l'environnement de stockage.



Dans cette fenêtre, vous devez saisir :

- Votre code utilisateur,
- Le code de la bibliothèque,
- Le type de session : session courante ou session historisée,
- Le numéro de session.

Les informations saisies dans cette fenêtre sont sauvegardées dans un fichier de code **vapback.ini**. Ce fichier est relu au moment de l'ouverture de la fenêtre.

Une fois la procédure locale exécutée, vous obtenez un fichier de mouvements prêt à être transféré dans le Référentiel à l'aide d'un utilitaire de transfert de fichiers.

Chaque objet VisualAge présent dans le source du fichier export correspond à un ensemble de mouvements de fichier de sauvegarde.

8.4.3. Sauvegarde centrale

La partie centrale du traitement de sauvegarde est composée de deux procédures batch : **VUP1** et **VUP2**. Les fonctionnalités, les entrées utilisateur et les résultats de chacune des procédures sont présentés ci-dessous.

8.4.3.1. Procédure VUP1 : Calcul des codes entités

8.4.3.1.1. Fonctionnalités

La procédure **VUP1** permet essentiellement de calculer les codes entités pour les objets VisualAge. Elle crée les éléments qui seront utilisés en entrée de la procédure **VUP2**. Elle utilise en entrée le fichier de sauvegarde produit par le traitement local de la procédure de sauvegarde et transféré sur site central.

Elle est constituée des étapes suivantes :

- Extraction des codes entités des objets VisualAge avec création d'un fichier de correspondance entre les codes entités et les identifiants VisualAge,
- Analyse du fichier de sauvegarde et calcul des codes entités pour les nouvelles entités créées dans VisualAge pour constituer un fichier de nouveaux codes.

A partir du fichier de sauvegarde produit par le traitement local de la sauvegarde, la procédure **VUP1** détermine les codes entités par les opérations suivantes :

- Conversion des caractères minuscules en majuscules,
- Remplacement/suppression des caractères spéciaux,
- Si l'identifiant VisualAge est composé de 6 caractères, recherche de l'existence du code entité correspondant. Ou récupération des 6 premiers caractères de l'identifiant VisualAge qui vont constituer le code entité,
- Vérification de l'existence de ce nouveau code dans la table de correspondance VisualAge Pacbase, puis dans la liste des codes déjà calculés.
- En cas de collision, incrémentation du dernier caractère selon la séquence alphanumérique. A la fin de la séquence, si le problème n'est pas résolu, itération du calcul sur l'avant dernier caractère, et ainsi de suite.

Exemple : pour un code égal à NOMCLI, on obtient la séquence calculée suivante : NOMCLI, NOMCLA, NOMCLB, ..., NOMCLZ, NOMCA, NOMCB ... , NOMCZ, NOMCAA, NOMCAB, NOMCAZ ...., NOMCBA ... etc...

Les codes calculés sont proposés à l'utilisateur. Ce dernier a la possibilité d'intervenir pour le confirmer ou le modifier.

Pour plus de détails concernant d'éventuelles modifications, consultez le paragraphe **8.4.3.2**.

8.4.3.1.2. Entrées utilisateur

Le fichier créé par le traitement local de la sauvegarde est composé d'une ligne d'assignation utilisateur (*) indiquant l'environnement VisualAge Pacbase de sauvegarde, suivie des lignes de description des objets VisualAge, objet par objet.

Vous devez compléter cette ligne par :

- le mot de passe
- le code Produit et le numéro d'Amélioration si la base est sous contrôle DSMS.

POSITION	LONGUEUR	VALEUR	SIGNIFICATION
02	02	*	Code ligne
04	08		Code Utilisateur
12	**08**		**Mot de passe**
20	03		Code Bibliothèque VisualAge Pacbase
23	05		Numéro et état de la session
		' '	Session courante
58	**09**		**Produit + numéro d'Amélioration si Base sous contrôle DSMS**

8.4.3.1.3. Résultats

Cette procédure produit trois fichiers :

- Fichier de correspondance entre les codes entités et les identifiants VisualAge (fichier VP),
- Fichier de nouveaux codes entités calculés pour les nouvelles entités créées dans VisualAge (fichier VV),
- Un fichier de mouvements correspondant au fichier produit par le traitement local de la procédure de sauvegarde, après suppression des doubles (fichier VG).

8.4.3.2. Procédure VUP2 : Génération des mouvements de mise à jour

8.4.3.2.1. Fonctionnalités

La procédure **VUP2** analyse le fichier de mouvements issu de **VUP1** et met à niveau la base en prenant en compte uniquement les différences entre le fichier de mouvements et la base. Elle crée un fichier de mouvements de mise à jour qui sera utilisé en entrée de la procédure batch **UPDT**.

8.4.3.2.2. Entrées utilisateur

La procédure **VUP2** utilise en entrée les trois fichiers issus de la procédure **VUP1**, en prenant en compte les éventuelles modifications effectuées par l'utilisateur dans le fichier des nouveaux codes (VN). Ce fichier est constitué d'une ligne d'assignation utilisateur (*) et de mouvements pour la mise à jour de la base.

La procédure **VUP2** comprend les deux types d'entrées utilisateur présentés ci-après.

- **Fichier de mouvements (VG)**

Ce fichier est constitué d'une ligne '*' et de lignes permettant de générer les mouvements de mise à jour de la base. La ligne '*' est obligatoire.

La ligne '*' est la suivante :

POSITION	LONGUEUR	VALEUR	SIGNIFICATION
03	01	*	Code ligne
12	08		Mot de passe
58	09		Produit + numéro Amélioration si base sous contrôle DSMS

- **Fichier des nouveaux codes (VN)**

Ce fichier comprend une ligne mouvement que vous pouvez modifier si vous souhaitez attribuer à une objet VisualAge un code entité différent de celui qui a été calculé automatiquement par **VUP1**.
Les modifications s'effectuent à l'aide d'un éditeur de texte.

La ligne mouvement est la suivante :

POSITION	LONGUEUR	VALEUR	SIGNIFICATION
55	06		Nouveau code de l'entité

8.4.3.2.3. Résultats

La procédure **VUP2** produit un fichier de mouvements de mise à jour qui sera utilisé en entrée de la procédure **UPDT**.

8.5. Restauration des données dans VisualAge Smalltalk

La procédure de Restauration permet de restituer dans votre station VisualAge les objets dont le source, produit par la fonctionnalité Export, a été sauvegardé au préalable dans le Référentiel VisualAge Pacbase.

L'unité de base de demande pour la Restauration est une Application VisualAge avec les Parts associés et les Applications filles.

La procédure de Restauration se compose de deux parties, l'une s'exécutant en central, l'autre en local.

8.5.1. Restauration centrale

8.5.1.1. Procédure VDWN

Avant d'exécuter cette procédure, il est nécessaire d'allouer les deux fichiers suivants :

- Fichier résultat de la Restauration centrale,
- Fichier des commandes de génération des composants Proxy extraits.

8.5.1.1.1. Fonctionnalités

La procédure **VDWN** permet de créer un fichier de mouvements qui sera importé dans votre station VisualAge.

8.5.1.1.2. Entrées utilisateur

La procédure **VDWN** comprend deux types d'entrées utilisateur.

- **Ligne définissant la bibliothèque-session à traiter**

POSITION	LONGUEUR	VALEUR	SIGNIFICATION
02	01	*	Code ligne
03	08		Code Utilisateur
11	08		Mot de passe
19	03		Code Bibliothèque
22	05		Numéro et état de la session
		' '	Session courante

- **Ligne de commande d'extraction**

POSITION	LONGUEUR	VALEUR	SIGNIFICATION
02	01	Y3	Code ligne
04	02		Classe de l'objet
		77	Application VisualAge
06	06		Code entité de l'objet VisualAge

8.5.1.1.3. Résultats

La procédure **VDWN** produit deux fichiers :

- Fichier central de restauration pour stocker les objets extraits du Référentiel. Ce fichier doit être transféré sur votre station VisualAge pour y être traité par la Restauration locale. Le fichier source ainsi produit sera importé dans VisualAge.
- Fichier de commandes de génération des composants Proxy s'ils sont utilisés dans les objets extraits. Il permet de regénérer les composants Proxy en cas de besoin. Ces commandes sont utilisables à l'entrée de la procédure **GPRT**.

8.5.2. Restauration locale dans VisualAge Pacbase for Smalltalk

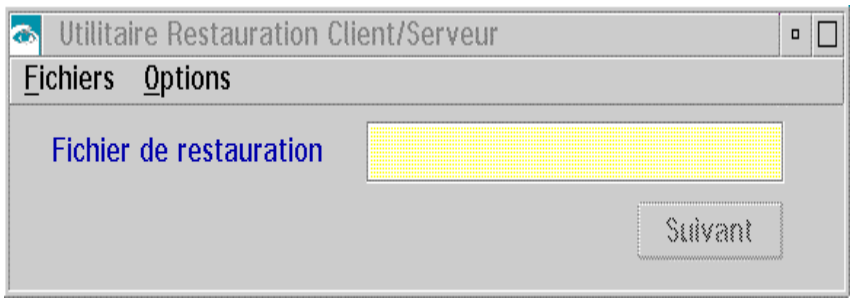
8.5.2.1. Fonctionnalités

Le traitement local de la procédure de Restauration consiste à reconstituer un fichier de description complet sous la forme d'un fichier source à partir du fichier de restauration produit par la procédure **VDWN**. Elle permet de préparer un fichier source propre à être importé dans VisualAge Smalltalk.

8.5.2.2. Accès à la fenêtre de Restauration Locale

Vous accédez à cette fenêtre à partir de la fenêtre **VisualAge Organizer**, menu **Outils**, choix **Restauration.**

La fenêtre suivante s'affiche :



8.5.2.3. Fenêtre Utilitaire Restauration Client/Serveur

Vous devez choisir le fichier en entrée. Il s'agit du fichier de mouvements produit par la procédure de Restauration centrale **VDWN** transféré sur votre poste à l'aide d'un utilitaire de transfert de fichiers.

La boîte d'édition Fichier de restauration permet de saisir le nom du fichier contenant les applications à restaurer. Vous pouvez également choisir l'option **Ouvrir** du menu **Fichiers** et sélectionner le fichier à traiter.

Il est fortement conseillé d'attribuer l'extension **.APP**, ce qui facilitera la gestion de vos fichiers.

Dans le menu **Options**, choisissez la langue dans laquelle s'afficheront les libellés des fenêtres .

Le bouton **Suivant** permet d'accéder à la fenêtre suivante.

Pour quitter le pont, cliquez sur le menu **Fichiers**, choix **Quitter.**

Une boîte d'édition affiche soit les messages d'erreur si un problème est rencontré dans le fichier de restauration et dans ce cas le bouton **OK** de la fenêtre d'erreur est grisé, soit la liste des applications contenues dans le fichier de restauration.

Le bouton **Restauration** permet de continuer la restauration des applications.

Le bouton Previous permet d'arrêter le processus de restauration et de revenir à la version précédente.

8.6. Epuration des entités VisualAge dans le Référentiel

8.6.1. Fonctionnalités

La procédure batch d'épuration **VPUR** permet de supprimer du Référentiel les occurrences d'entités VisualAge non utilisées. Elle consiste à chercher dans le réseau de bibliothèques ces entités. Sont considérées comme non utilisées les occurrences suivantes :

- les Parts qui n'appartiennent à aucune Application,
- les Applications qui ne présentent ni Application archive, ni Application mère, ni Application fille.

Il est possible de limiter le champ des recherches en spécifiant une liste donnée de codes bibliothèques et de numéros de sessions.

8.6.2. Entrées utilisateur

- **Ligne d'identification utilisateur**

POSITION	LONGUEUR	VALEUR	SIGNIFICATION
02	01	*	Code ligne
03	08		Code utilisateur
11	08		Mot de passe

- **Lignes de sélection des bibliothèques**

Si aucune ligne n'est spécifiée, la recherche s'effectue dans toutes les bibliothèques.

POSITION	LONGUEUR	VALEUR	SIGNIFICATION
02	02	'SL'	Code ligne
04	03		Code de la bibliothèque sélectionnée

- **Lignes de sélection des sessions**

Si aucune ligne n'est spécifiée, la recherche s'effectue dans toutes les sessions, y compris la session courante.

POSITION	LONGUEUR	VALEUR	SIGNIFICATION
02	02	'SS'	Code ligne
04	03		Code et état de la session (session courante : 9999Z)

8.6.3. Résultats

A l'issue de la procédure **VPUR**, un fichier de mouvements à utiliser en entrée de la procédure de mise à jour **UPDT** est produit.

9. Annexe 1 : Paramétrage des logiciels externes

Un paramétrage spécifique des logiciels externes est nécessaire à la mise en oeuvre des communications pour une application Pacbench Client/Serveur.

Les fiches de configuration qui suivent correspondent à des configurations testées. Elles sont donc dépendantes d'un environnement technique particulier. Par exemple pour les communications entre les plateformes OS/2 ou Windows 3.1/95/NT et le host MVS, les postes de travail du réseau Token-Ring s'adressent à une Gateway SNA Communication Manager ou SNA Server, elles-mêmes s'adressant à un contrôleur IBM 3745 pour atteindre le host MVS.

Elles ne proposent donc qu'une solution de paramétrage pour un contexte particulier et doivent être adaptées aux impératifs techniques de chaque site.

9.1. IMS CPI-C

9.1.1. Configuration MVS et IMS

9.1.1.1. Définitions VTAM

Minima requis : VTAM Version 3.3

9.1.1.1.1. Définition de l'ATCSTR de VTAM

```
*******************************
NOPROMPT,CONFIG=00,SSCPID=01,
MAXSUBA=31,SUPP=NOSUP,
SSCPNAME=A01M,
SSCPORD=DEFINED,
NETID=NETCGI,
HOSTSA=1,
CRPLBUF=(550,,20,,40,40),
IOBUF=(420,182,25,,40,40),
LFBUF=(300,,0,,20,10),
LPBUF=(50,,0,,5,5),
SFBUF=(50,,0,,5,5),
SPBUF=(90,,0,,5,5),
NOTRACE,TYPE=VTAM
*******************************
```

9.1.1.1.2. MAC Address du TIC (attachement LAN) contrôleur 3745 dans NCP :

```
************************************************
*         TIC BNN                              05742090
************************************************
A01L1TK  LINE ADDRESS=(1088,FULL),
            PORTADD=01,
            LOCADD=400003172000,
            ISTATUS=ACTIVE,
                  UACB=(X$P1AX,X$P1AR)
*
A01TK1PU PU ISTATUS=ACTIVE,
            ADDR=01
************************************************
```

9.1.1.1.3. Définition APPC/IMS

Une application spécifique APPC/IMS pour LU6.2 doit être définie dans VTAM (différente de l'APPL IMS utilisée pour les terminaux 3270).

```
*/* LIB: SYS1.VTAMLST(A0100)
*/*
A01IMS62 APPL     EAS=50,                 ESTIMATED CONCURRENT SESSIONS   *
                  MODETAB=MTLU62,         ⇨      nom de la table des modes
                  DLOGMOD=LU62,                   ⇨      nom du mode dans la table
                  APPC=YES,               ⇨      paramètre obligatoire
                  ACBNAME=A01IMS62,       APPLID FOR ACB
                  AUTH=(ACQ,BLOCK,PASS)   IMS CAN ACQUIRE & PASS TMLS
```

9.1.1.1.4. Définition du mode

- Définition des caractéristiques pour les Sessions LU6.2
- Le Mode SNASVCMG est utilisé avec le support "Parallels Sessions".

```
TITLE '--- "MODTABLE" CONCERNANT LES LU 6.2 ---'
*
MTLU62   MODETAB
         SPACE 4
SNASVCMG MODEENT LOGMODE=SNASVCMG, FMPROF=X'13',
               TSPROF=X'07', PRIPROT=X'B0', SECPROT=X'B0',
               COMPROT=X'D0B1', RUSIZES=X'8585', ENCR=B'0000',
               PSERVIC=X'060200000000000000000300'
LU62     MODEENT LOGMODE=LU62, TYPE=X'00', FMPROF=X'13',
               TSPROF=X'07', PRIPROT=X'B0', SECPROT=X'B0',
               COMPROT=X'50B1', RUSIZES=X'8787', SRCVPAC=X'00',
               PSNDPAC=X'00', SSNDPAC=X'00',
PSERVIC=X'06020000000000000002C00'
```

9.1.1.1.5. Définition de la passerelle SNA

```
*/* LIB: SYS1.VTAMLST(SW1TKR)
*/*
*/*
*/* SWITCHED MAJOR NODE TOKEN-RING ST-MARC :            -  06/07/95.
*/*   --->    LIEN XCA MAJOR NODE ==> XCA1TKR  (IBM3172-3)
*/*   --->    LIEN GROUPE XCA     ==> GRP02
*             ------------------------------
*    - MODEL FOR IDBLK X'05D' - OS/2 COMMUNICATIONS MANAGER
*             ------------------------------
*
SW1TKR   VBUILD TYPE=SWNET,MAXNO=99,MAXGRP=10
*
*          ----------------------------------------
* ------->  DEFINING A GATEWAY TOKEN-RING --> GTWTK1  <--------
*          ----------------------------------------
W1TK00   PU    ADDR=50,
               CPNAME=GTWTK1,
               IDBLK=05D,
               IDNUM=00002,
               DYNLU=YES,
               MAXPATH=1,
               DISCNT=NO,
               IRETRY=YES,
```

```
               VPACING=7,
               PACING=7,
               SSCPFM=USSSCS,
               MAXDATA=4096,
               PUTYPE=2
```

L'option DYNLU=YES permet d'éviter toute définition complémentaire de Lu 6.2 au niveau VTAM (pour les postes du réseau TR).

9.1.1.1.6. Définition d'une LU indépendante

Malgré ce qui vient d'être dit ci-dessus, il peut être intéressant de définir une LU indépendante pour les premiers tests de communication :

```
*/* LIB: SYS1.VTAMLST(SW1TKR)
*/*
*/*
*/* SWITCHED MAJOR NODE TOKEN-RING ST-MARC :            -  06/07/95.
*/*
*/*
*  INDEPENDENT LUS
*
IMS4349           LU    LOCADDR=0,
                  ISTATUS=ACTIVE,
                  DLOGMOD=LU62,
                  MODETAB=MTLU62
```

9.1.1.2. Définitions APPC/MVS

Minima requis : MVS/ESA Version 4.2

Deux membres de la SYS1.PARMLIB sont nécessaires pour définir les caractéristiques de la Local LU APPC/MVS et d'ASCH (scheduler APPC).

9.1.1.2.1. Définition de la Local LU APPC/MVS

- Paramétrage

```
BROWSE -- SYS1.PARMLIB(APPCPM00) - 01.12 ----------- LINE 0000
 COMMAND ===>
******************************** TOP OF DATA ****************
/*************************************************************/
/*  THE FOLLOWING PARAMETERS ARE FOR THE APPC ADDRESS SPACE. */
/*  THE APPC ADDRESS SPACE HANDLES THE ACTUAL COMMUNICATIONS.*/
/*                                                           */
/*  THESE MEMBERS PROVIDE THE LINKAGE BETWEEN LU NAMES AND   */
/*  TRANSACTION SCHEDULERS.                                  */
/*************************************************************/
LUADD
      ACBNAME(A01IMS62)            ⇨ correspond à l'APPL APPC/IMS de VTAM
      SCHED(CGIB)
      BASE
      TPDATA(UTI.APPCTP)
      TPLEVEL(SYSTEM)
SIDEINFO DATASET(UTI.APPCSI)
```

- Lancement

```
BROWSE -- SYS1.PROCLIB(APPC) ----------------------
 COMMAND ===>
******************************** TOP OF DATA ********
//APPC PROC APPC=00
//APPC EXEC PGM=ATBINITM,PARM='APPC=&APPC',REGION=0K
```

```
****************************** BOTTOM OF DATA ******
```

9.1.1.2.2. Définition du scheduler APPC/MVS

- Paramétrage

```
BROWSE -- SYS1.PARMLIB(ASCHPM00) - 01.00 -----------
 COMMAND ===>
******************************** TOP OF DATA ********
/*****************************************************
/* THE FOLLOWING IS ADDED TO ENABLE APPC/MVS.
/*****************************************************
 CLASSADD CLASSNAME(SVSAMP)
      MAX(10)
      MIN(2)
      RESPGOAL(0.02)
      MSGLIMIT(700)
 OPTIONS DEFAULT(SVSAMP)
      SUBSYS(JES2)
 TPDEFAULT REGION(4M)
      TIME(10,30)
      MSGLEVEL(1,1)
      OUTCLASS(R)
****************************** BOTTOM OF DATA ******
```

- Lancement

```
BROWSE -- SYS1.PROCLIB(ASCH) ----------------------
 COMMAND ===>
******************************** TOP OF DATA ********
//ASCH PROC ASCH=00
//ASCH EXEC PGM=ASBSCHIN,PARM='ASCH=&ASCH',REGION=0K
****************************** BOTTOM OF DATA ******
```

9.1.1.3. Définitions IMS

Minima requis : IMS Version 4.1

9.1.1.3.1. IMSCTRL Macro

IMS doit être généré en utilisant les bibliothèques MVS/ESA correspondant à la version spécifiée en troisième paramètre du mot-clé `SYSTEM`. Cette version MVS/ESA doit être au minimum la 4.2.

```
BROWSE -- EX.IMS410.SOURCE(STAGE1) - 01.28 -------------
 COMMAND ===>
*
* IMSCTRL MACRO --
*
         IMSCTRL  SYSTEM=(VS/2,CTLBLKS,4.2),
               DBRC=(YES,NO),
               DBRCNM=DBRC41,
               DLINM=DLISAS41,
               DCLWA=YES,
               IMSID=CGIB,          ⇨ correspond au paramètre SCHED de APPCPM00
               NAMECHK=(YES,S1),
               MAXIO=(,015),
               MAXREGN=(008,512K,A,A),
               MCS=(8),
               DESC=7,
               MAXCLAS=020
```

9.1.1.3.2. Startup

Pour permettre à IMS d'établir la connexion avec APPC/MVS, il faut spécifier `APPC=Y` dans le Job de Startup IMS (`DFSPBxxx` où `xxx` est le suffixe de la région).

Définition de la transaction

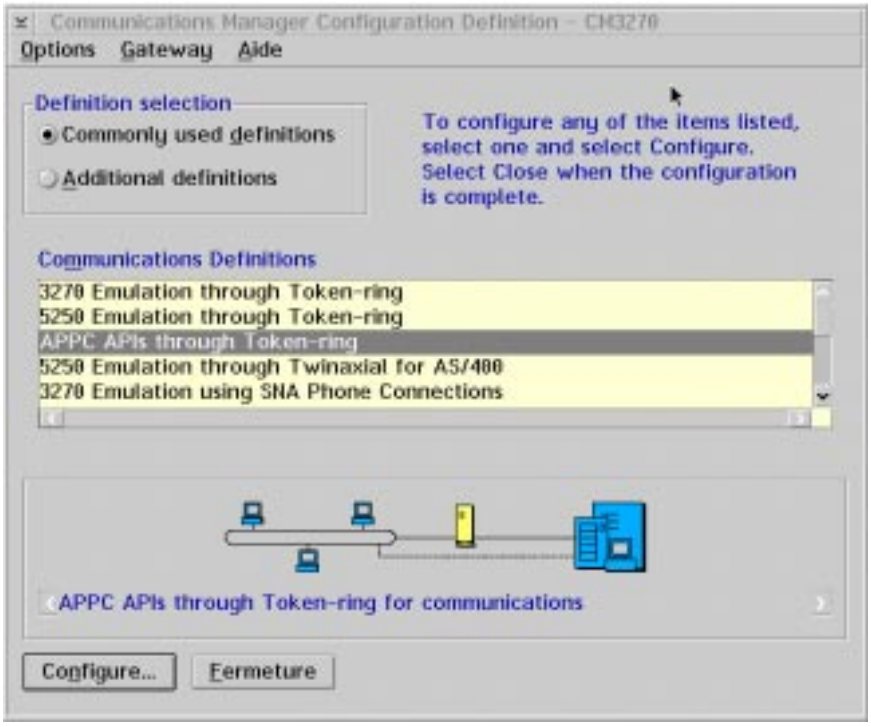
```
BROWSE -- EX.IMS410.SOURCE(STAGE1) - 01.28 ---------
 COMMAND ===>
******** TRANSACTION DB2 POUR BABY CLIENT (PTAB) ********************
APPLCTN  PSB=BABIVG
TRANSACT CODE=BABI,SEGSIZE=00000,MODE=SNGL,SEGNO=00000,
              PRTY=(07,10,00002),PROCLIM=(00005,00015),EDIT=ULC,
              MSGTYPE=(SNGLSEG,RESPONSE,4)
```

9.1.2. Configuration OS/2

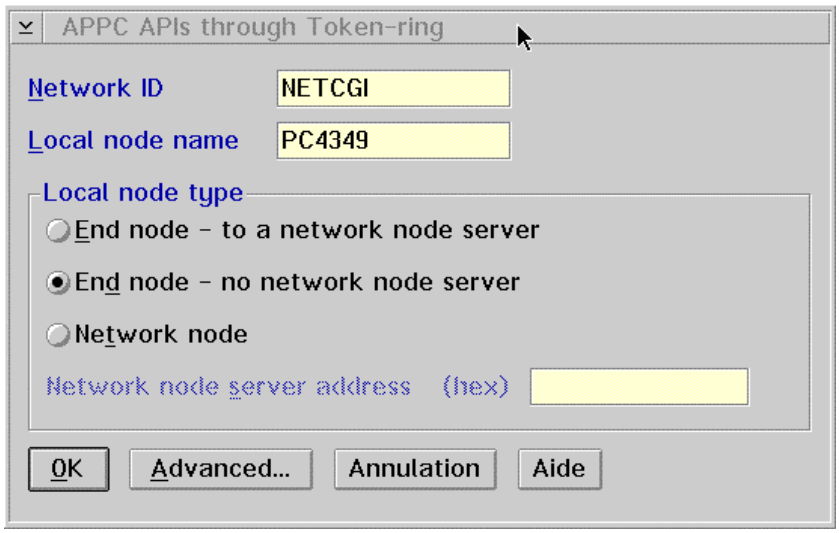
9.1.2.1. Communication Manager/2 : configuration APPC

Minima requis : Communication Manager/2 Version 1.11

9.1.2.1.1. Choix de la communication

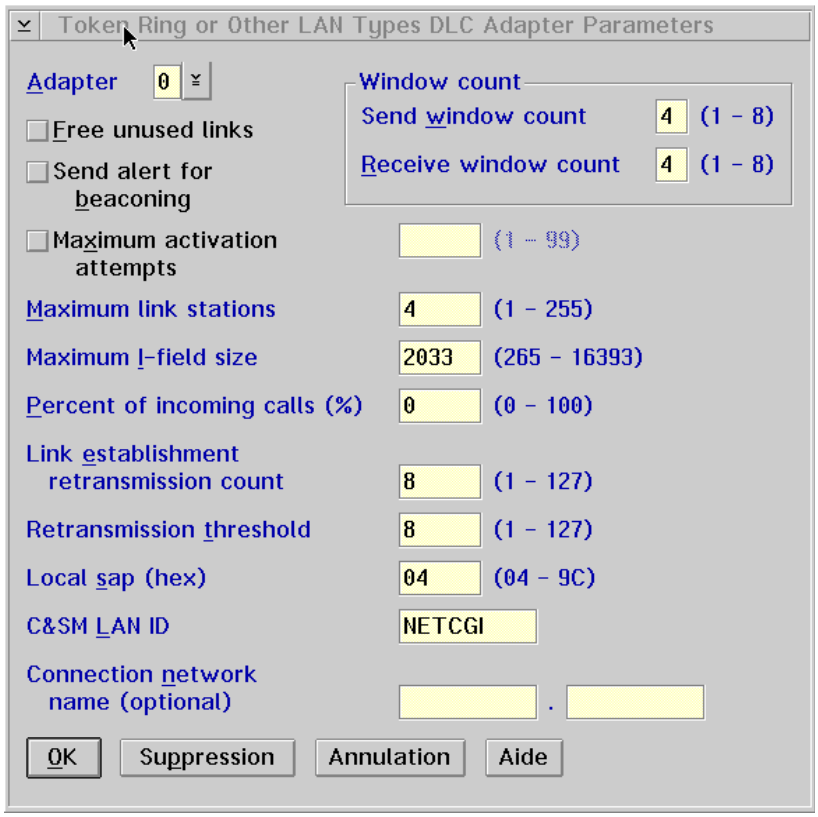


9.1.2.1.2. Réseau Token Ring



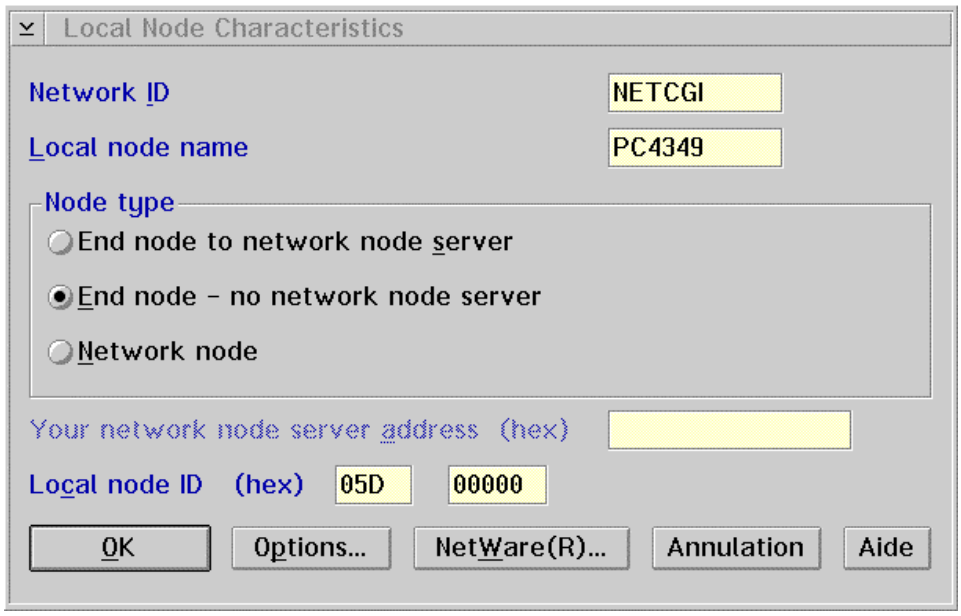
`Network ID` : Identifiant du réseau SNA sur lequel le poste est connecté (`NETID` de VTAM)
`Local node name` : identification du poste de travail dans le réseau

9.1.2.1.3. Token Ring DLC Adapter

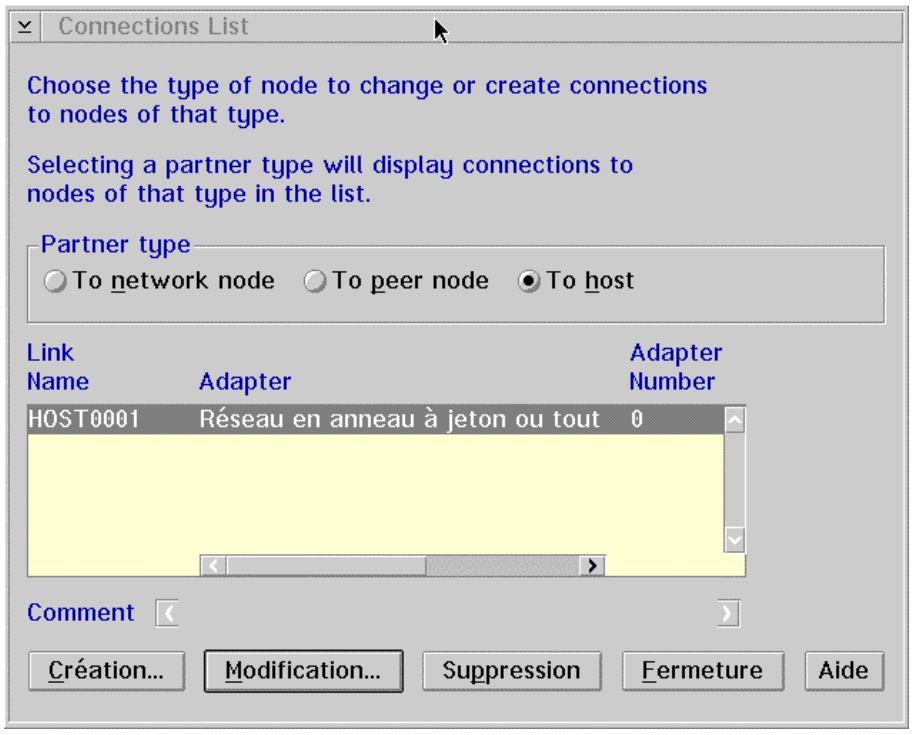


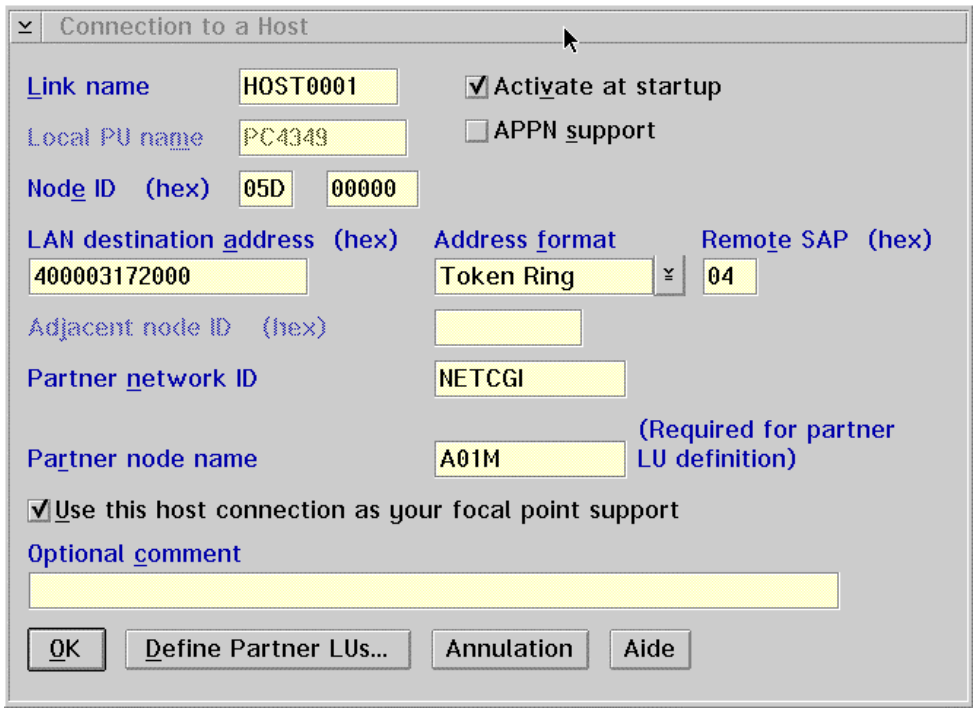
`Maximum I-field size` : Cette valeur dépend de la passerelle CM/2.

9.1.2.1.4. Local Node



9.1.2.1.5. Connections List

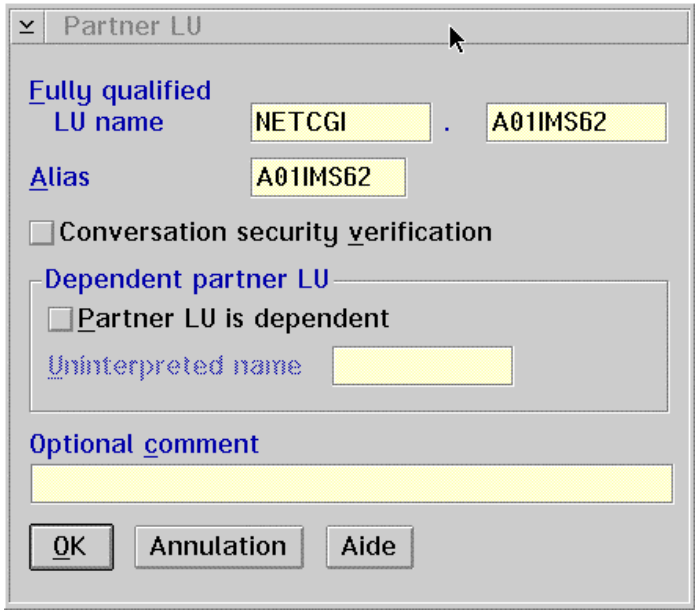




`Node ID` : n'est pas utilisé dans notre environnement car on s'adresse à la Gateway CM/2 via sa MAC Address
`LAN destination address`: MAC Address de la Gateway CM/2
`Partner Node Name` : nom du Noeud VTAM

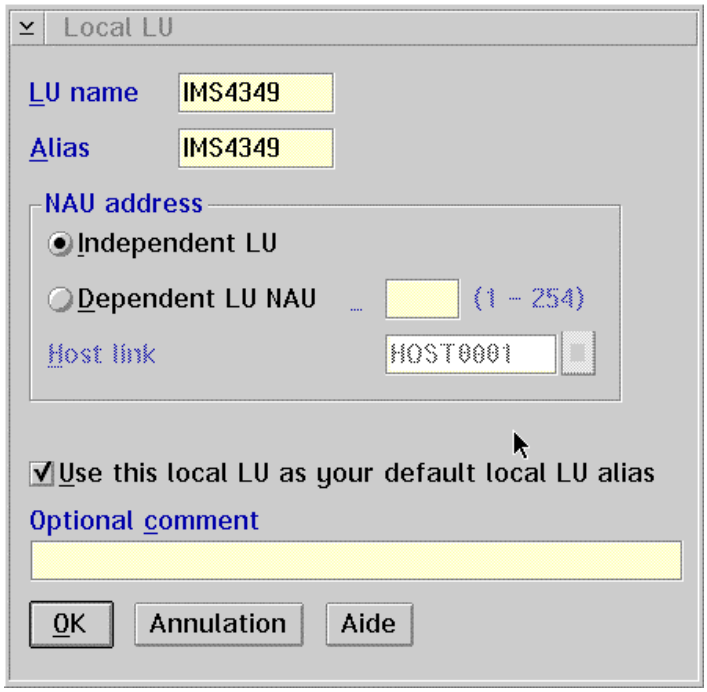
9.1.2.1.6. Partner LU

La LU Partenaire doit être créée à partir du bouton poussoir `Define Partner LUs...` de l'écran précédent.



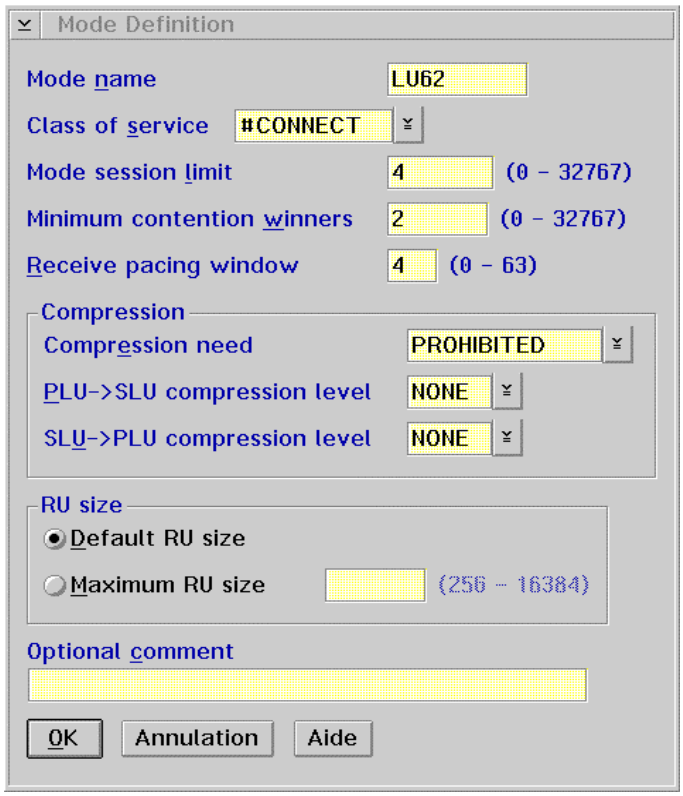
`LU Name` : paramètre `NETID` de la définition générale de VTAM (NETCGI)
+ paramètre `APPL` de définition d'IMS dans la VTAM List (`A01IMS62`).

9.1.2.1.7. Local LU



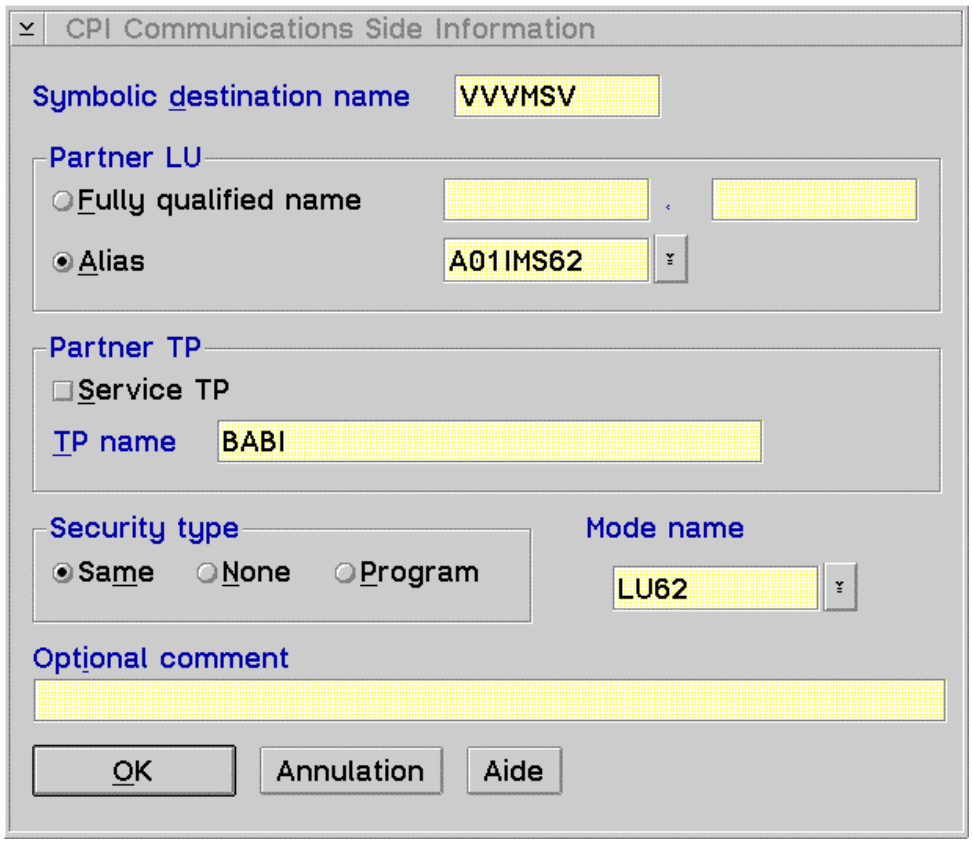
`LU Name` : aucune correspondance sur le Host si option Dynamic LU (`DYNLU=YES`)

9.1.2.1.8. Mode



`Mode Name` : mode défini dans la table des modes VTAM, paramètre `LOGMODE` de la macro-instruction `MODEENT`

9.1.2.1.9. Side Info



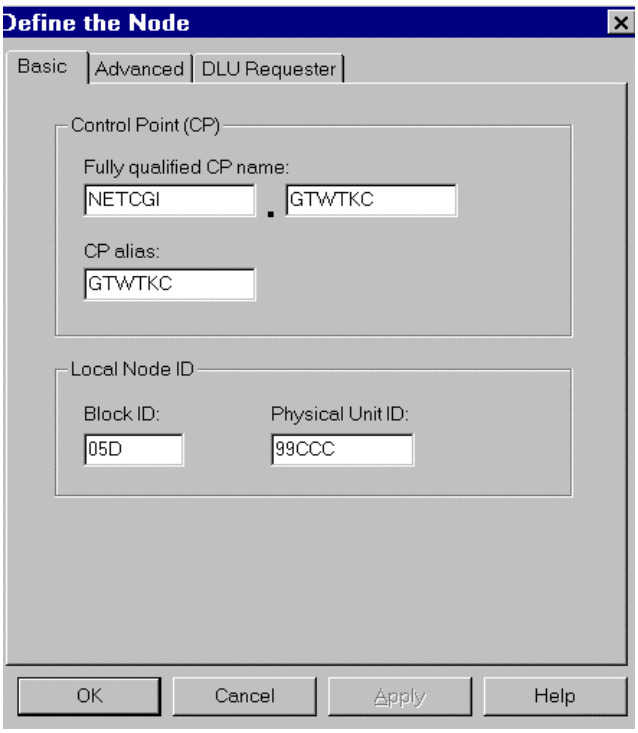
Cet écran est nécessaire pour les communications CPI-C afin d'établir la connexion entre les clients et les programmes serveur utilisant APPC.
Le `Symbolic destination name` doit correspondre au nom externe du moniteur de communication dans le Référentiel.
`TP name` : code de la transaction activant le moniteur de communication sur le Host.

9.1.3. Configuration WINDOWS 95/NT

9.1.3.1. Personal Communications : configuration APPC

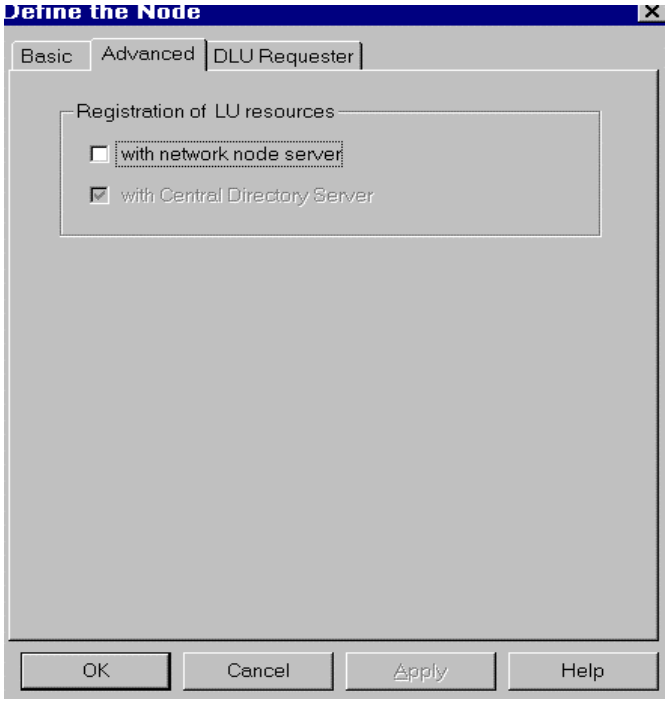
Minima requis : Personal Communications Version 4.2

9.1.3.1.1. Define the Node

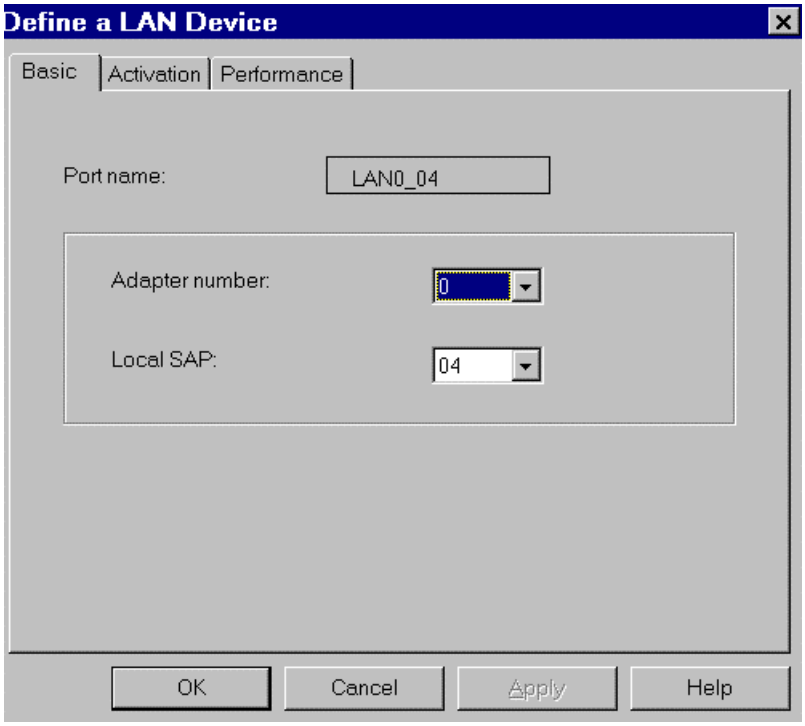


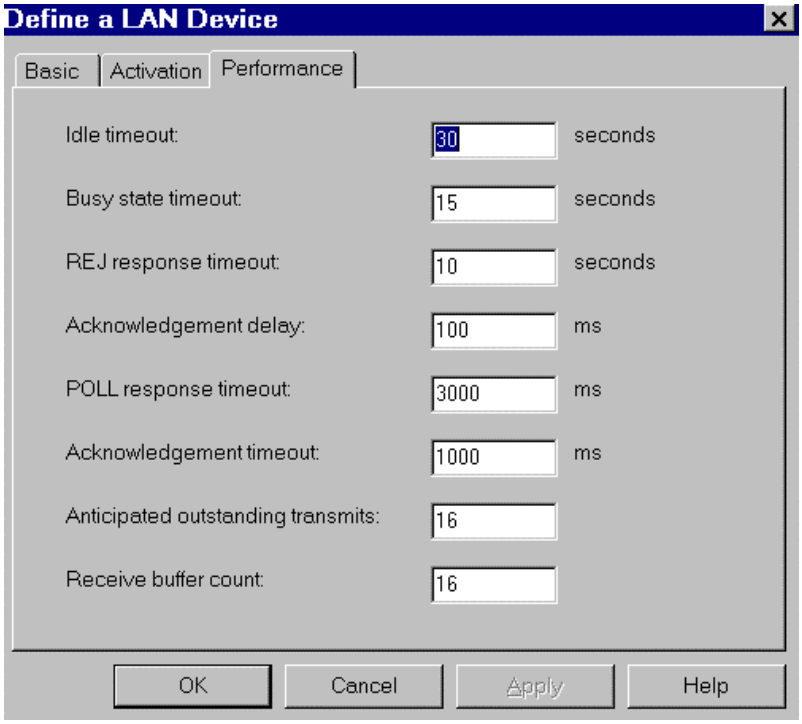
`Network ID` : Identifiant du réseau SNA sur lequel le poste est connecté (`NETID` de VTAM)

`Local node name` : identification du poste de travail dans le réseau

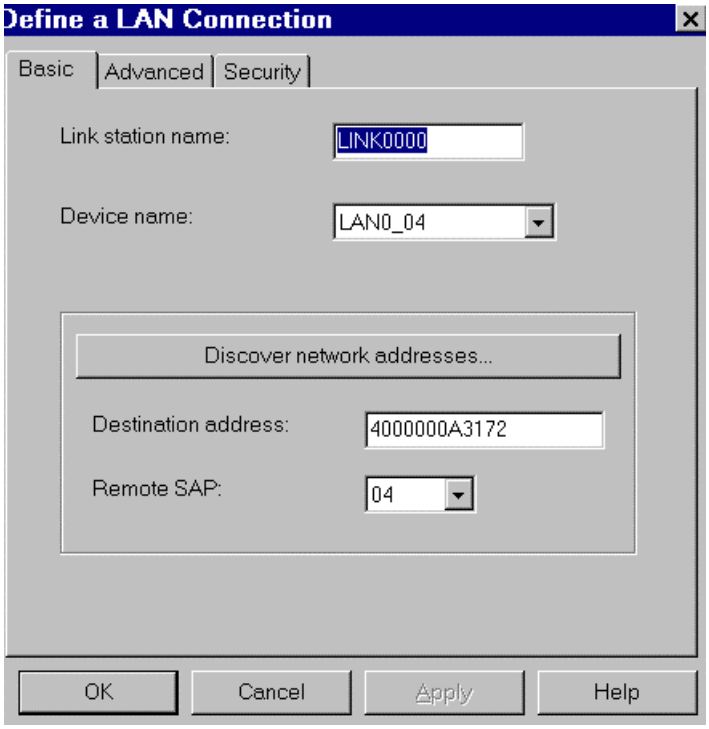


9.1.3.1.2. Define the Lan Device

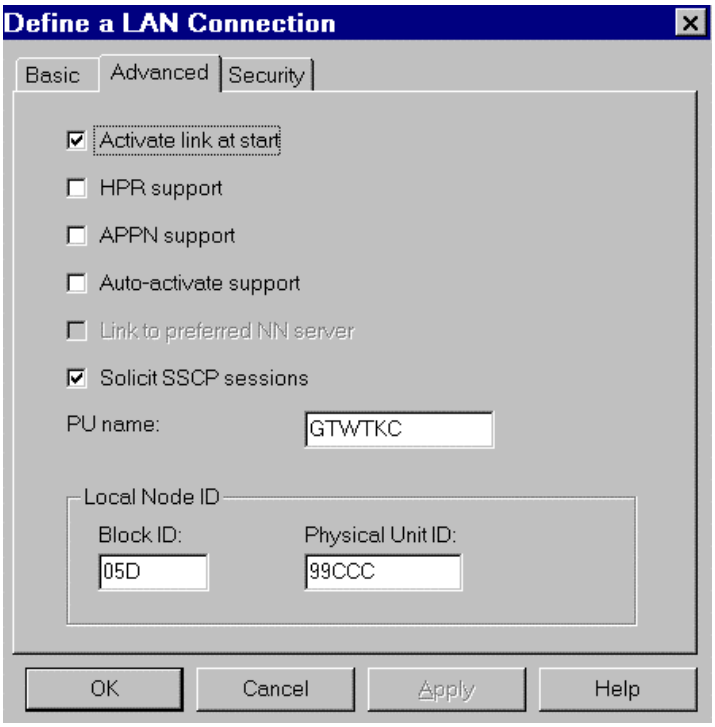


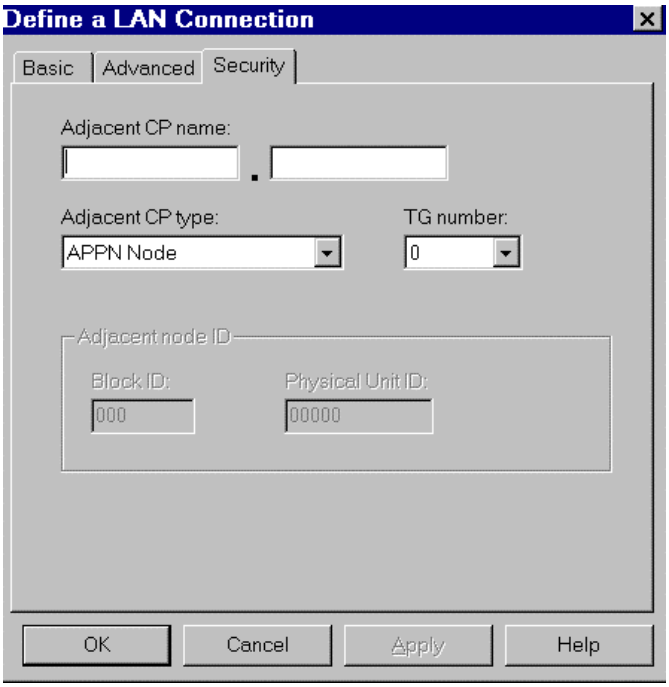


9.1.3.1.3. Define the Lan Connection

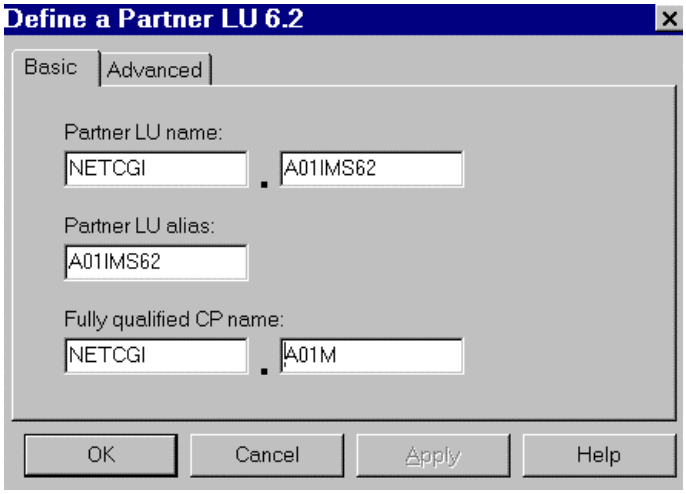


`Destination address:` MAC Address du TIC IBM 3745


Define a LAN Connection
Basic
Advanced
Security
Activate link at start
HPR support
APPN support
Auto-activate support
Link to preferred NN server
Solicit SSCP sessions
PU name:
GTWTKC
Local Node ID
Block ID:
05D
Physical Unit ID:
99CCC
OK
Cancel
Apply
Help

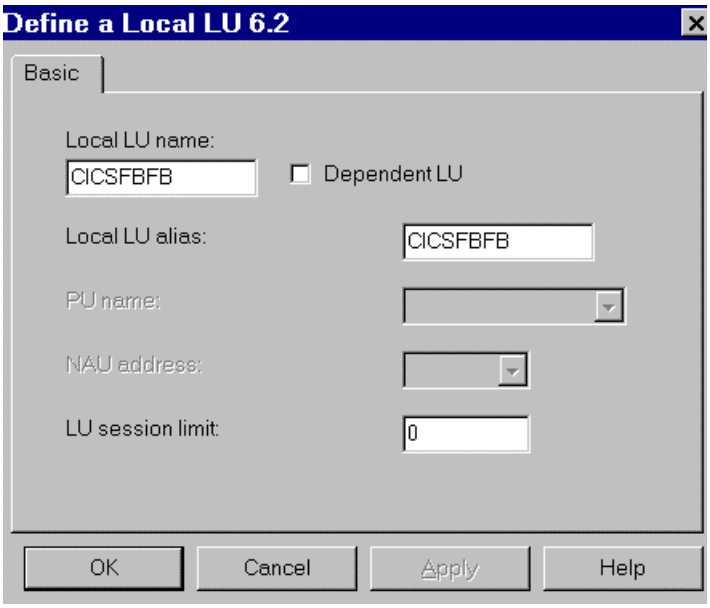

Define a LAN Connection
Basic
Advanced
Security
Adjacent CP name:
Adjacent CP type:
APPN Node
TG number:
0
Adjacent node ID
Block ID:
000
Physical Unit ID:
00000
OK
Cancel
Apply
Help

9.1.3.1.4. Define a Partner LU 6.2



`LU Name` : paramètre `NETID` de la définition générale de VTAM (NETCGI) + paramètre `APPL` de définition d'IMS dans la VTAMLST (`A01IMS62`).

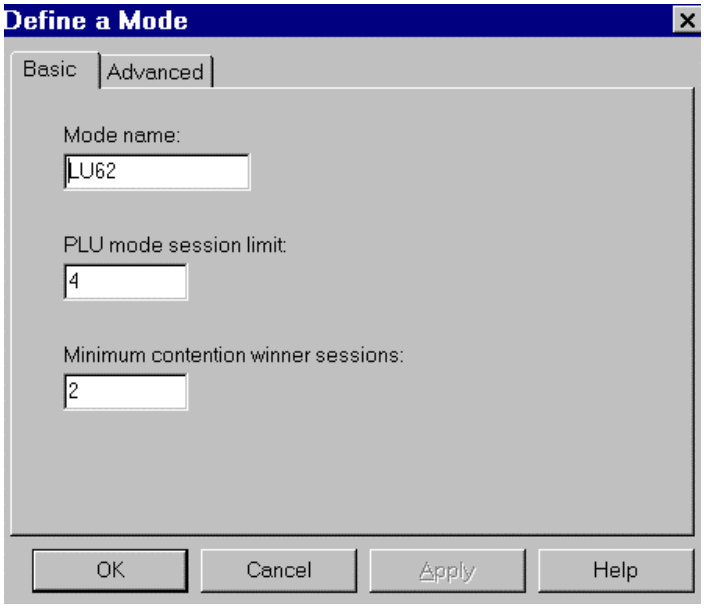
9.1.3.1.5. Define a Local LU 6.2



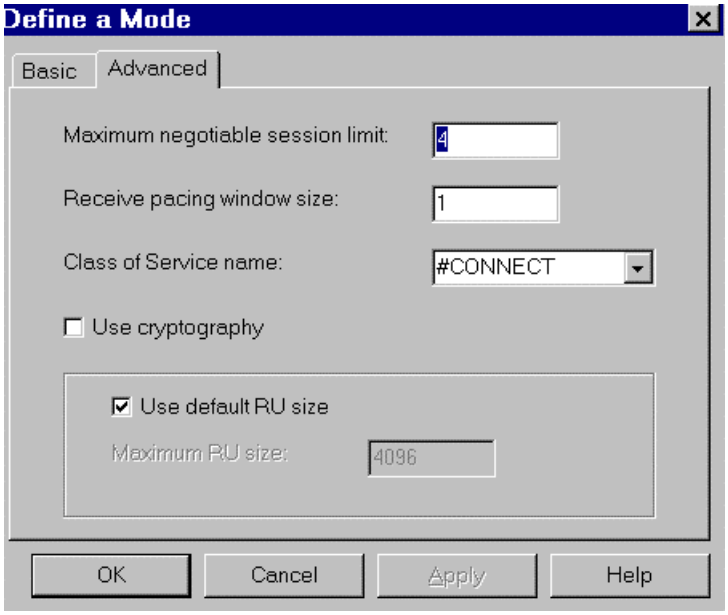
`LU Name` : aucune correspondance sur le Host si option Dynamic LU (`DYNLU=YES`)

Dans les variables d'environnement Windows 95 definir la locale LU6.2 utilisée par défaut => SET APPCLLU=CICSFBFB

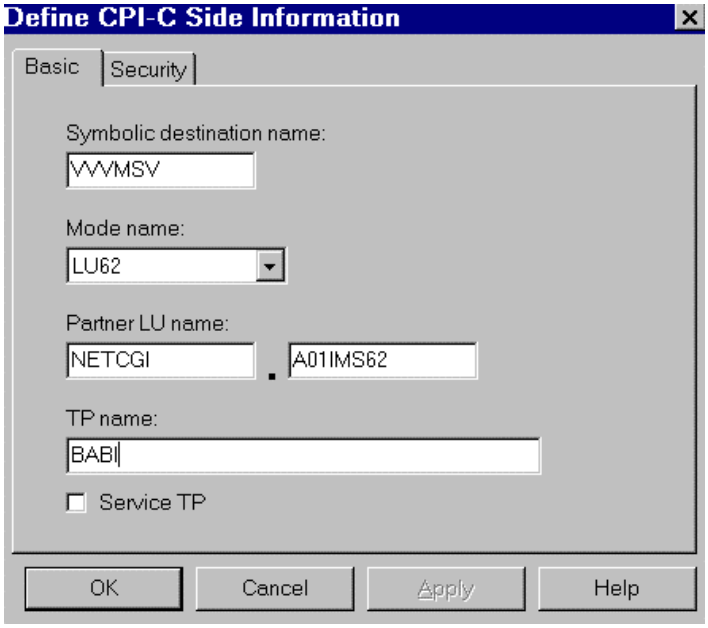
9.1.3.1.6. Define a Mode



`Mode Name` : mode défini dans la table des modes VTAM, paramètre `LOGMODE` de la macro-instruction `MODEENT`



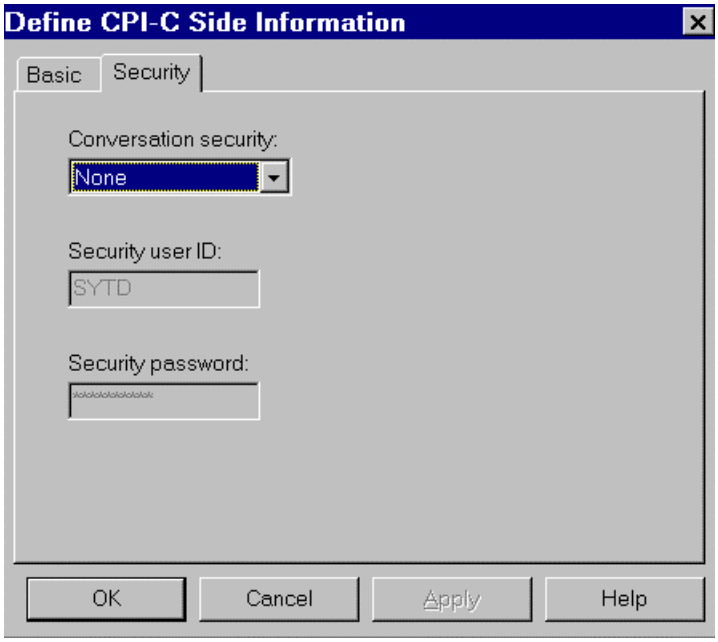
9.1.3.1.7. Define CPI-C Side Information



Cet écran est nécessaire pour les communications CPI-C afin d'établir la connexion entre les clients et les programmes serveur utilisant APPC.

Le `Symbolic destination name` doit correspondre au nom externe du moniteur de communication dans le Référentiel.

`TP name` : code de la transaction activant le moniteur de communication sur le Host.



9.2. CICS CPI-C

9.2.1. Configuration MVS et CICS

9.2.1.1. Définitions VTAM

Minima requis : VTAM Version 3.3

9.2.1.1.1. Définition de l'ATCSTR de VTAM

```
*******************************
NOPROMPT,CONFIG=00,SSCPID=01,
MAXSUBA=31,SUPP=NOSUP,
SSCPNAME=A01M,
SSCPORD=DEFINED,
NETID=NETCGI,
HOSTSA=1,
CRPLBUF=(550,,20,,40,40),
IOBUF=(420,182,25,,40,40),
LFBUF=(300,,0,,20,10),
LPBUF=(50,,0,,5,5),
SFBUF=(50,,0,,5,5),
SPBUF=(90,,0,,5,5),
NOTRACE,TYPE=VTAM
*******************************
```

9.2.1.1.2. MAC Address du TIC (attachement LAN) contrôleur 3745 dans NCP :

```
************************************************
*         TIC BNN                          05742090
************************************************
A01L1TK  LINE ADDRESS=(1088,FULL),
              PORTADD=01,
              LOCADD=400003172000,
              ISTATUS=ACTIVE,
                      UACB=(X$P1AX,X$P1AR)
*
A01TK1PU PU ISTATUS=ACTIVE,
              ADDR=01
************************************************
```

9.2.1.1.3. Définition du CICS

```
A01CICS1 APPL  EAS=160,                ESTIMATED CONCURRENT SESSIONS   *
               ACBNAME=CICST,          APPLID FOR ACB                  *
               AUTH=(ACQ,VPACE,PASS),  CICS CAN ACQUIRE & PASS TMLS    *
*                                      CICS CAN REQUEST BLOCKED INPUT

               PARSESS=YES                ⇨    Supports parallel Sessions
               SONSCIP=YES,                                             *
               MODETAB=MTLU62             ⇨    nom de la table des modes
```

9.2.1.1.4. Définition du mode

- Définition des caractéristiques pour les Sessions LU6.2

- Le Mode SNASVCMG est utilisé avec le support "Parallels Sessions".

```
TITLE '--- "MODTABLE" CONCERNANT LES LU 6.2 ---'
*
MTLU62   MODETAB
         SPACE 4
SNASVCMG MODEENT LOGMODE=SNASVCMG, FMPROF=X'13',
               TSPROF=X'07', PRIPROT=X'B0', SECPROT=X'B0',
               COMPROT=X'D0B1', RUSIZES=X'8585', ENCR=B'0000',
               PSERVIC=X'060200000000000000000300'
LU62     MODEENT LOGMODE=LU62, TYPE=X'00', FMPROF=X'13',
               TSPROF=X'07', PRIPROT=X'B0', SECPROT=X'B0',
               COMPROT=X'50B1', RUSIZES=X'8787', SRCVPAC=X'00',
               PSNDPAC=X'00', SSNDPAC=X'00',
PSERVIC=X'060200000000000000002C00'
```

9.2.1.1.5. Définition de la passerelle SNA

```
*/* LIB: SYS1.VTAMLST(SW1TKR)
*/*
*/*
*/* SWITCHED MAJOR NODE TOKEN-RING ST-MARC :            -  06/07/95.
*/*   --->    LIEN XCA MAJOR NODE ==> XCA1TKR  (IBM3172-3)
*/*   --->    LIEN GROUPE XCA     ==> GRP02
*             ------------------------------
*    - MODEL FOR IDBLK X'05D' - OS/2 COMMUNICATIONS MANAGER
*             ------------------------------
*
SW1TKR   VBUILD TYPE=SWNET,MAXNO=99,MAXGRP=10
*
*          ----------------------------------------
* -------> DEFINING A GATEWAY TOKEN-RING -->  GTWK1  <--------
*          ----------------------------------------
W1TK00   PU    ADDR=50,
               CPNAME=GTWTK1,
               IDBLK=05D,
               IDNUM=00002,
               DYNLU=YES,
               MAXPATH=1,
               DISCNT=NO,
               IRETRY=YES,
               VPACING=7,
               PACING=7,
               SSCPFM=USSSCS,
               MAXDATA=4096,
               PUTYPE=2
```

L'option `DYNLU=YES` permet d'éviter toute définition complémentaire de Lu 6.2 au niveau VTAM (pour les postes du réseau TR).

9.2.1.1.6. Définition d'une LU indépendante

Malgré ce qui vient d'être dit ci-dessus, il peut être intéressant de définir une LU indépendante pour les premiers tests de communication :

```
*/* LIB: SYS1.VTAMLST(SW1TKR)
*/*
*/*
*/* SWITCHED MAJOR NODE TOKEN-RING ST-MARC :            -  06/07/95.
*/*
*/*
*  INDEPENDENT LUS
*
CICSFBFB    LU    LOCADDR=0,
                  ISTATUS=ACTIVE,
                  DLOGMOD=LU62,
                  MODETAB=MTLU62
```

9.2.1.2. Définitions APPC/MVS

Minima requis : MVS/ESA Version 4.2

Il n'y a pas de définitions spécifiques car on utilise la couche APPC livrée avec la version de CICS.

9.2.1.3. Définitions CICS

9.2.1.3.1. Paramètre de l'InterSystem Communication dans la table SIT

ISC=YES

9.2.1.3.2. Connexion

```
  Connection     : SGFB
  Group          : GRPISC5
  DEscription    :
 CONNECTION IDENTIFIERS
  Netname        : CICSFBFB   ⇨      à déclarer comme Local LU dans CM/2 (codification
libre)
  INDsys         :
 REMOTE ATTRIBUTES
  REMOTESystem   :
  REMOTEName     :
 CONNECTION PROPERTIES
  ACcessmethod   : Vtam
  Protocol       : Appc
  SInglesess     : No         ⇨      dans le cas d'une LU indépendante
  DAtastream     : User
  RECordformat   : U
 OPERATIONAL PROPERTIES
  AUtoconnect    : Yes
  INService      : Yes
 SECURITY
  SEcurityname   : PTPD
  ATtachsec      : Verify     ⇨      Verification du userid et password à la conversation
  BINDPassword   :
  BINDSecurity   : No
```

9.2.1.3.3. Session

```
 Sessions       : SESSIOFB
 Group          : GRPISC5
 DEscription    :
SESSION IDENTIFIERS
 Connection     : SGFB            ⇨   code de la connexion définie ci-dessus
 SESSName       :
 NETnameq       :
 MOdename       : LU62            ⇨   mode défini dans la MODTABLE de VTAM
SESSION PROPERTIES
 Protocol       : Appc
 MAximum        : 004 , 002
 RECEIVEPfx     :
 RECEIVECount   :
 SENDPfx        :
 SENDCount      :
 SENDSize       : 08192
 RECEIVESize    : 08192
 SESSPriority   : 000
 Transaction    :
OPERATOR DEFAULTS
 OPERId         :
 OPERPriority   : 000
 OPERRsl        : 0
 OPERSecurity   : 1
PRESET SECURITY
 USERId         :
OPERATIONAL PROPERTIES
 Autoconnect    : Yes
 INservice      :
 Buildchain     : Yes
 USERArealen    : 000
 IOarealen      : 00000 , 00000
 RELreq         : Yes
 DIscreq        : No
 NEPclass       : 000
RECOVERY
 RECOVOption    : Sysdefault
 RECOVNotify    : None
```

9.2.1.3.4. Définition de la transaction

```
OBJECT CHARACTERISTICS                                    CICS RELEASE =
0330
   CEDB  View
    TRansaction    : VIC0                 ⇨  code à reporter dans le TP name du
                                             CPIC Side Information de CM/2
    Group          : VISUAL
    DEscription    :
    PROGram        : AVVMSV               ⇨  code du moniteur de communication
    TWasize        : 00000              0-32767
    PROFile        : DFHCICST
    PArtitionset   :
    STatus         : Enabled            Enabled ! Disabled
    PRIMedsize     : 00000              0-65520
    TASKDATALoc    : Below              Below ! Any
    TASKDATAKey    : User               User ! Cics
   REMOTE ATTRIBUTES
    DYnamic        : No                 No ! Yes
    REMOTESystem   :
    REMOTEName     :
    TRProf         :
    Localq         :                    No ! Yes
   SCHEDULING
    PRIOrity       : 001                0-255
    TClass         : No                 No ! 1-10
   ALIASES
    Alias          :
    TASKReq        :
    XTRanid        :
    TPName         :
                   :
    XTPname        :
                   :
                   :
    RECOVERY
    DTimout        : No                 No ! 1-6800
    Indoubt        : Backout            Backout ! Commit ! Wait
    RESTart        : No                 No ! Yes
    SPurge         : Yes                No ! Yes
    TPUrge         : Yes                No ! Yes
    DUmp           : Yes                Yes ! No
    TRACe          : Yes                Yes ! No
   SECURITY
    RESSec         : No                 No ! Yes
    Cmdsec         : No                 No ! Yes
    Extsec         : No                 No ! Yes
    TRANsec        : 01                 1-64
    RSl            : 00                 0-24 ! Public
```

De plus, si la transaction fait des accès à une base DB2, il faut que celle-ci soit autorisée et qu'elle soit rattachée au plan DB2. Le code transaction et le plan DB2 doivent donc être déclarés dans la Table RCT. Dans cet exemple, la transaction VIC0 et le plan DB2 ATDF sont utilisés par l'application Cliente :

```
DSNRCT TYPE = ENTRY,TXID=(VIC0),
       THRDM=6,THRDA=6,PLAN=ATDF,AUTH=(USERID,*,*)
```

9.2.2. Configuration OS/2

9.2.2.1. Communication Manager/2 : configuration APPC

Minima requis : Communication Manager/2 Version 1.11

9.2.2.1.1. Choix de la communication

Référez-vous à l'écran contenu dans le paragraphe équivalent du sous-chapitre 9.1, *IMS CPI-C*.

9.2.2.1.2. Réseau Token Ring

Référez-vous à l'écran contenu dans le paragraphe équivalent du sous-chapitre 9.1, *IMS CPI-C*.

9.2.2.1.3. Token Ring DLC Adapter

Référez-vous à l'écran contenu dans le paragraphe équivalent du sous-chapitre 9.1, *IMS CPI-C*.

9.2.2.1.4. Local Node

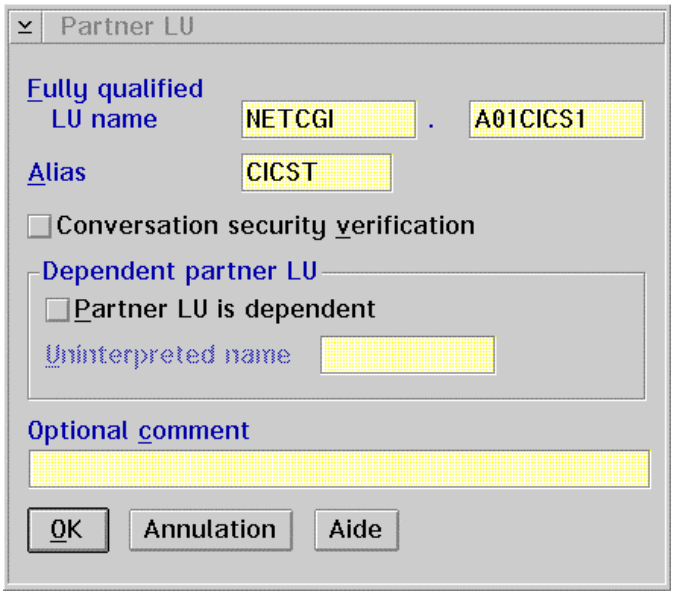
Référez-vous à l'écran contenu dans le paragraphe équivalent du sous-chapitre 9.1, *IMS CPI-C*.

9.2.2.1.5. Connections List

Référez-vous à l'écran contenu dans le paragraphe équivalent du sous-chapitre 9.1, *IMS CPI-C*.

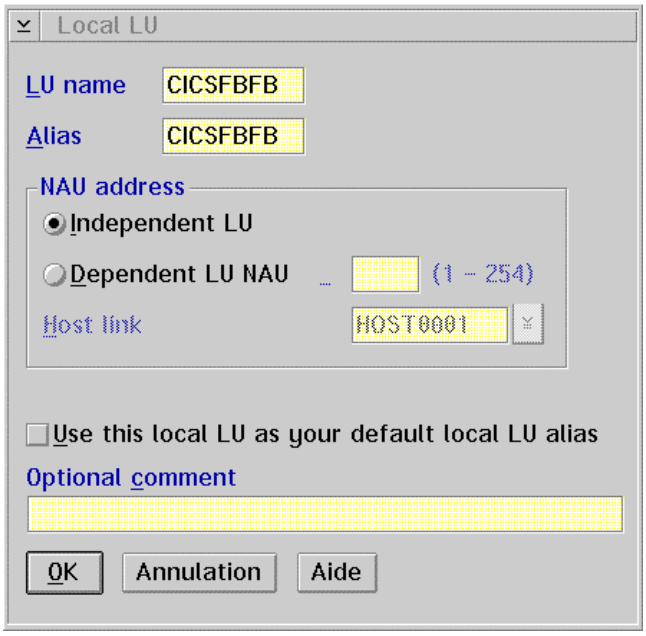
9.2.2.1.6. Partner LU

La LU Partenaire doit être créée à partir du bouton poussoir `Define Partner Lus...` de l'écran précédent.



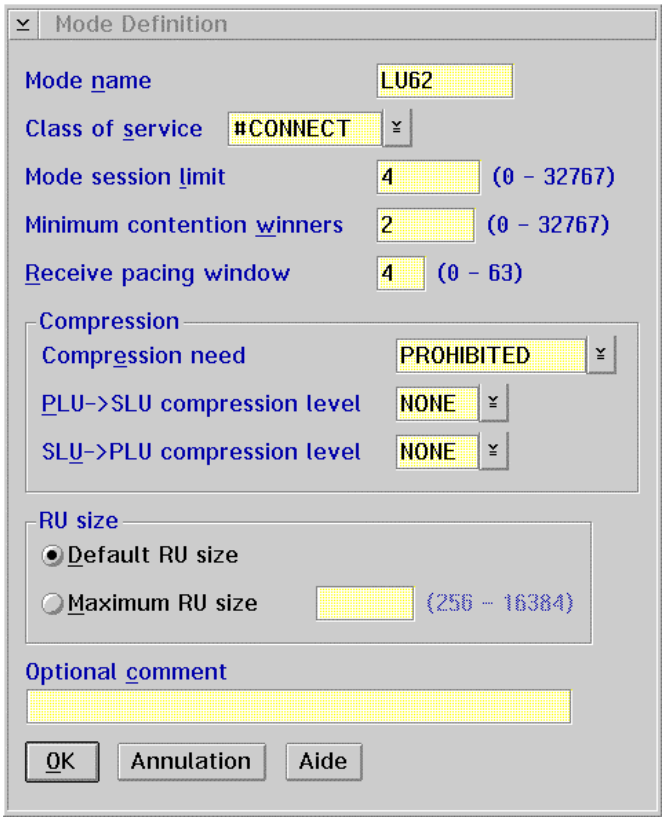
`LU Name` : paramètre `NETID` de la définition générale de VTAM (NETCGI)
+ paramètre `APPL` de définition du CICS dans la VTAM List (`A01CICS1`).

9.2.2.1.7. Local LU



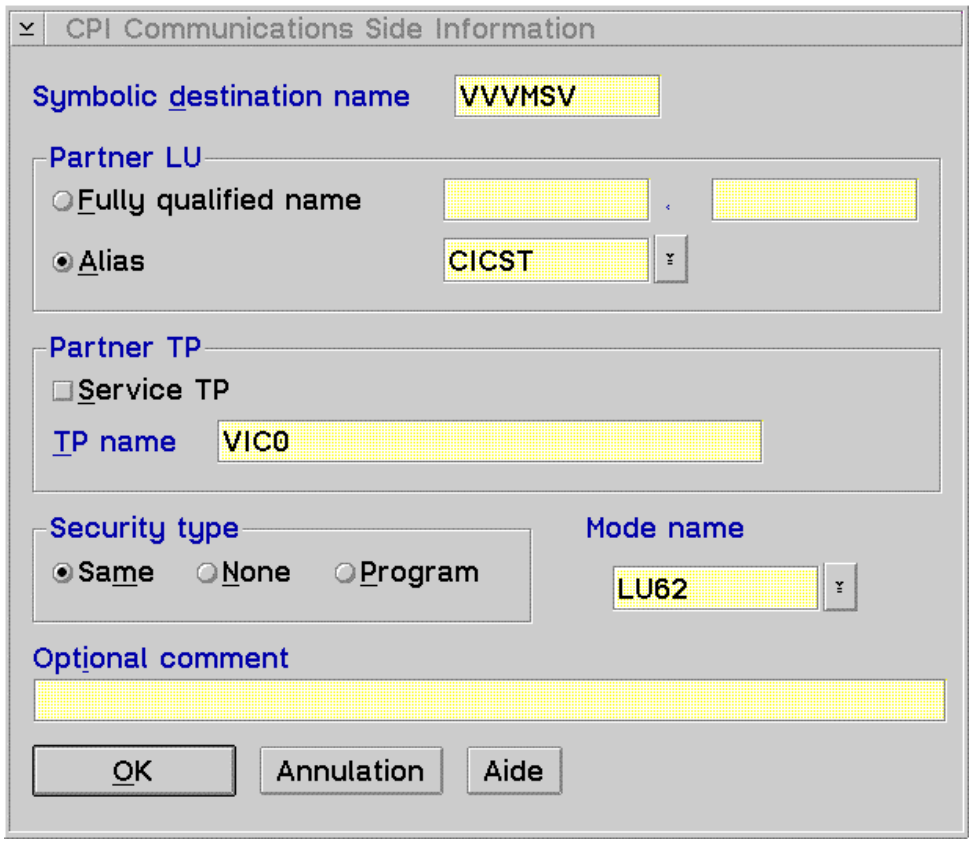
`LU Name` : Cette Local Lu doit être déclarée au niveau CICS Host dans le Netname de la connexion.

9.2.2.1.8. Mode



`Mode Name` : mode défini dans la table des modes VTAM, paramètre `LOGMODE` de la macro-instruction `MODEENT`

9.2.2.1.9. Side Info



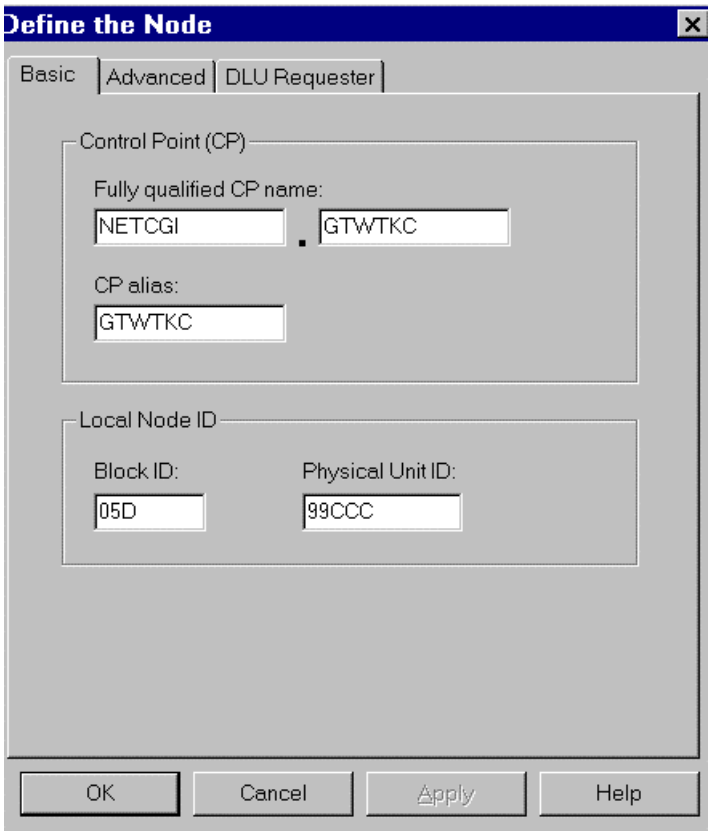
Cet écran est nécessaire pour les communications CPI-C afin d'établir la connexion entre les clients et les programmes serveur utilisant APPC.
Le `Symbolic destination name` doit correspondre au nom externe du moniteur de communication dans le Référentiel.
`TP name` : code de la transaction activant le moniteur de communication sur le Host.

9.2.3. Configuration WINDOWS 95/NT

9.2.3.1. Personal Communications : configuration APPC

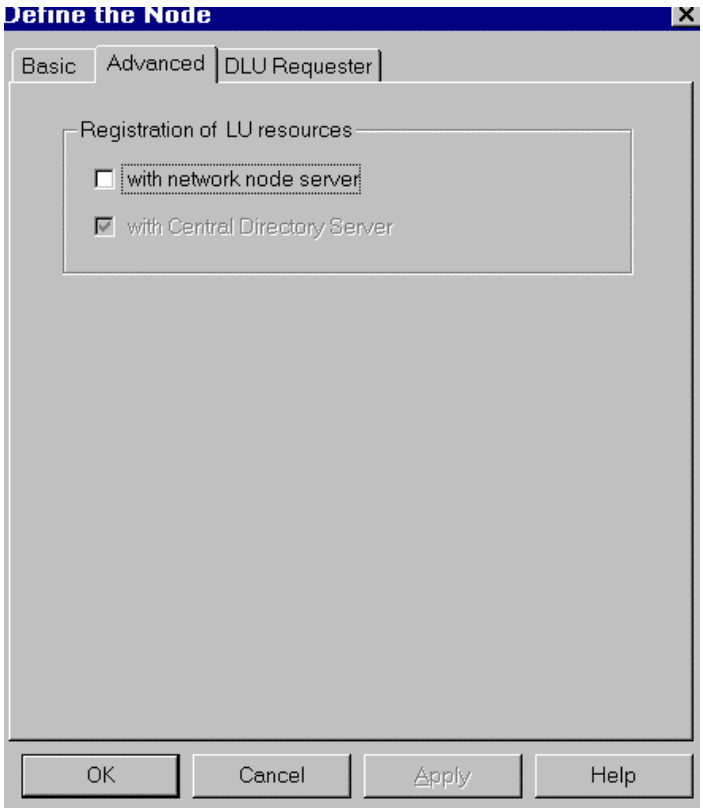
Minima requis : Personal Communications Version 4.2

9.2.3.1.1. Define the Node

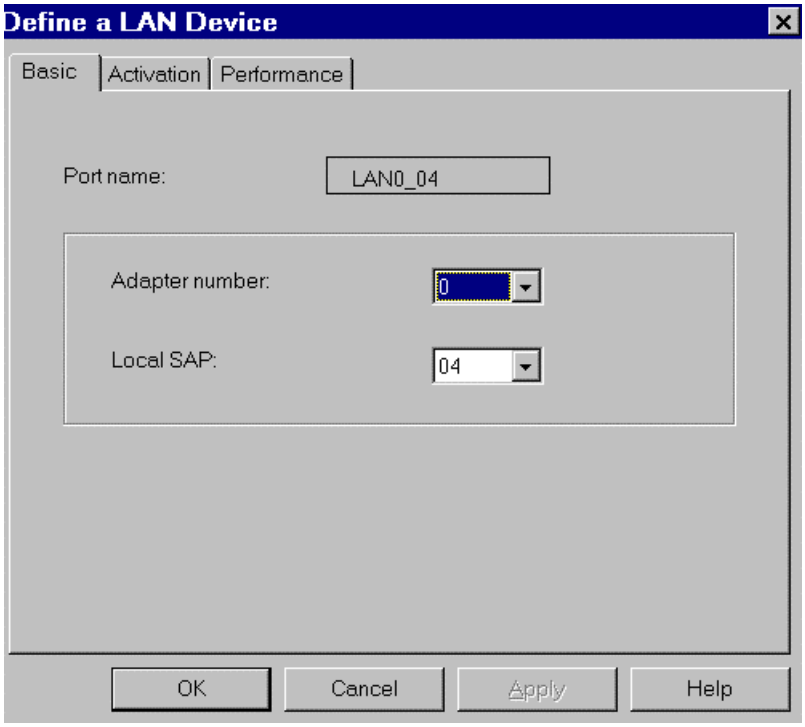


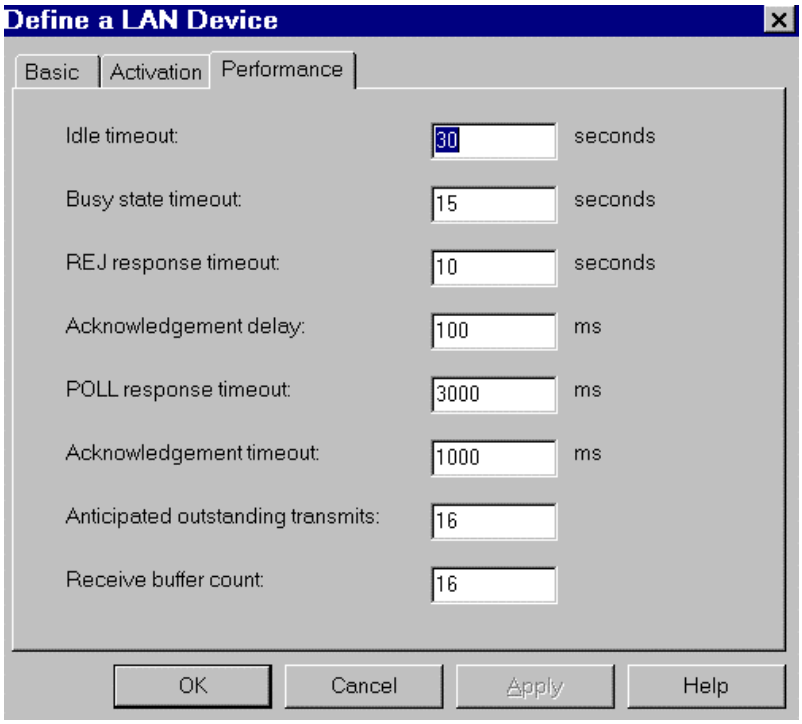
`Network ID` : Identifiant du réseau SNA sur lequel le poste est connecté (`NETID` de VTAM)

`Local node name` : identification du poste de travail dans le réseau

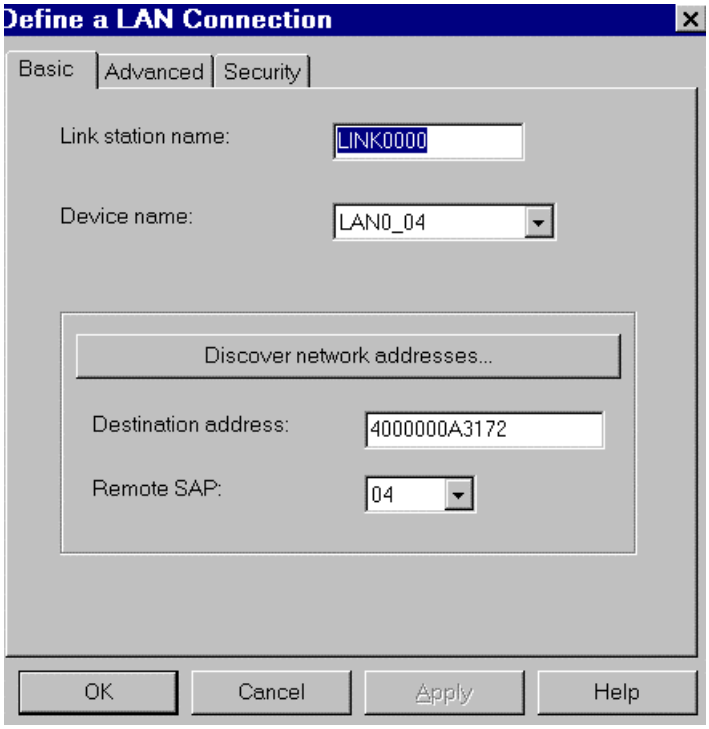


9.2.3.1.2. Define the Lan Device

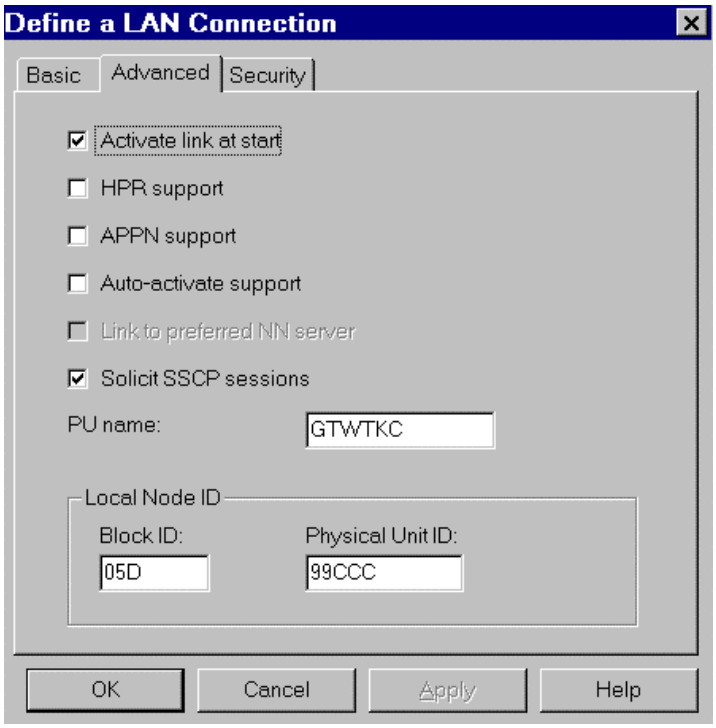


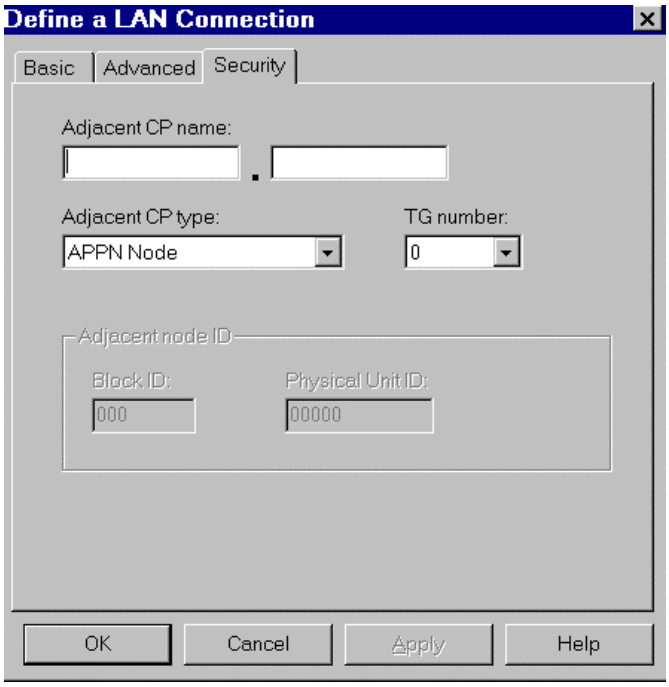


9.2.3.1.3. Define the Lan Connection

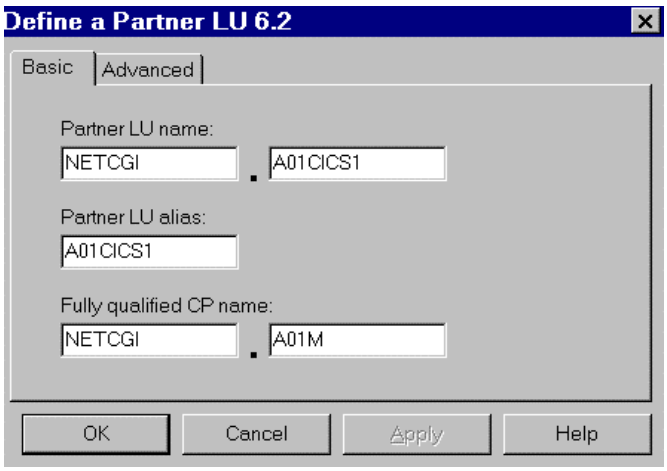


`Destination address:` MAC Address du TIC IBM 3745



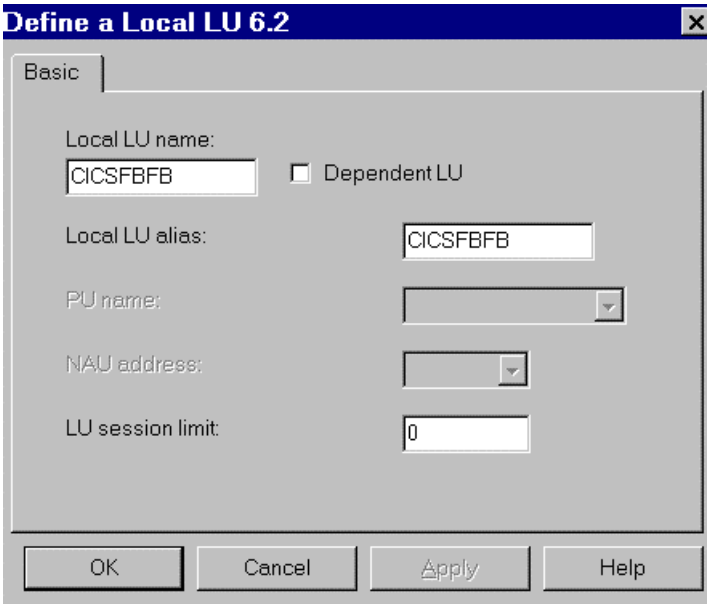


9.2.3.1.4. Define a Partner LU 6.2



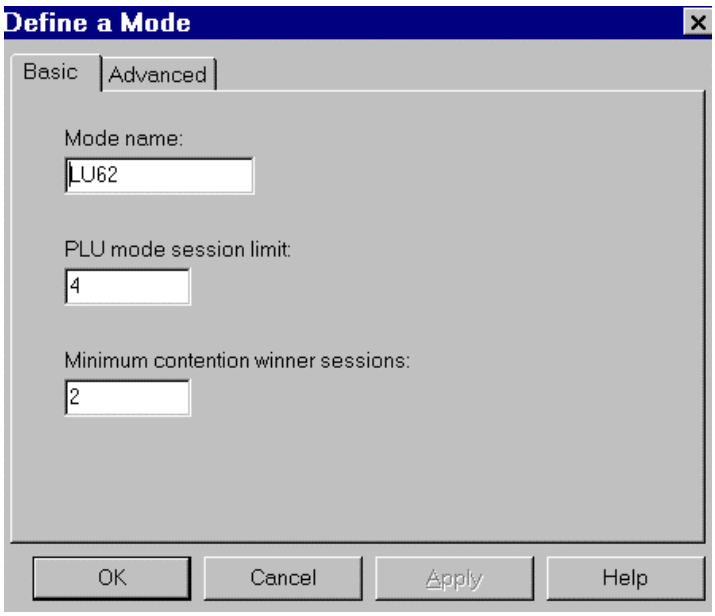
`LU Name` : paramètre `NETID` de la définition générale de VTAM (NETCGI) + paramètre `APPL` de définition du CICS dans la VTAMLST (`A01CICS1`).

9.2.3.1.5. Define a Local LU 6.2

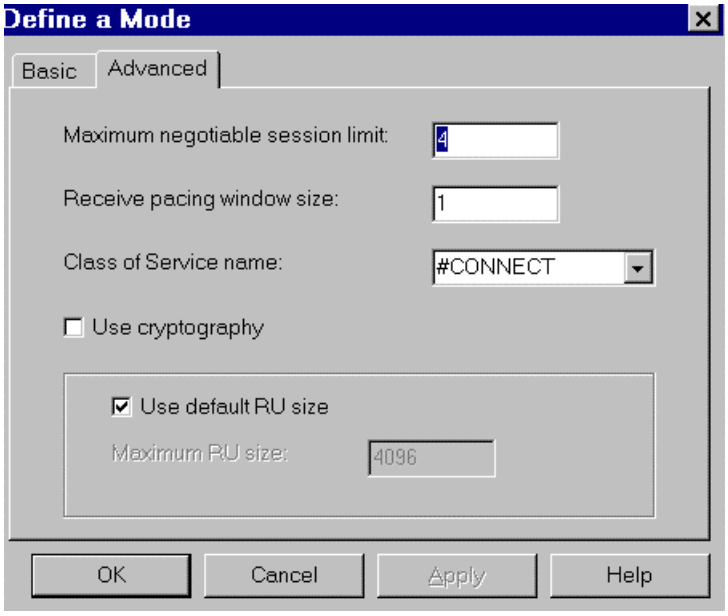


`LU Name` : aucune correspondance sur le Host si option Dynamic LU (`DYNLU=YES`)
Dans les variables d'environnement Windows 95 definir la locale LU6.2 utilisée par défaut => SET APPCLLU=CICSFBFB

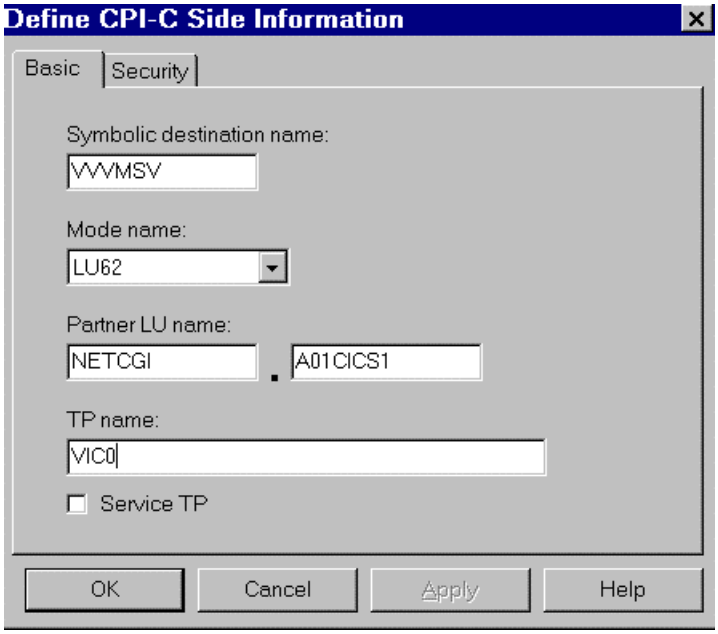
9.2.3.1.6. Define a Mode



`Mode Name` : mode défini dans la table des modes VTAM, paramètre `LOGMODE` de la macro-instruction `MODEENT`



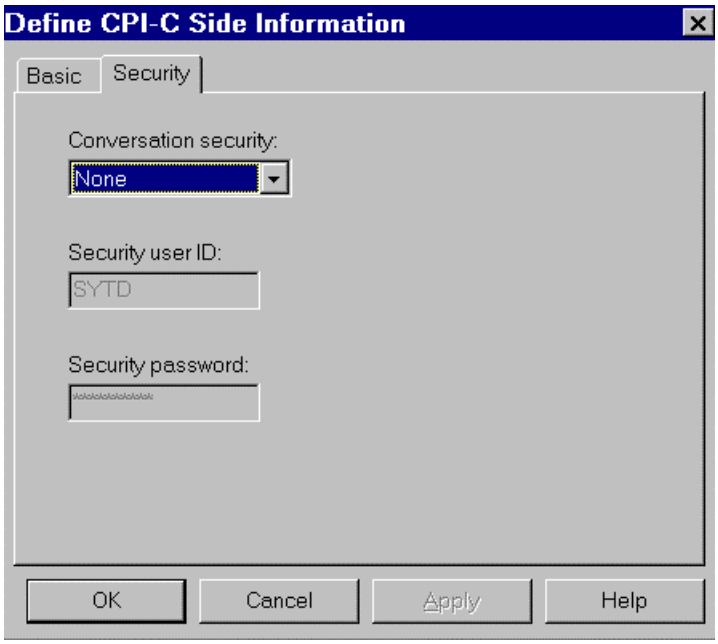
9.2.3.1.7. Define CPI-C Side Information



Cet écran est nécessaire pour les communications CPI-C afin d'établir la connexion entre les clients et les programmes serveur utilisant APPC.

Le `Symbolic destination name` doit correspondre au nom externe du moniteur de communication dans le Référentiel.

`TP name` : code de la transaction activant le moniteur de communication sur le Host.

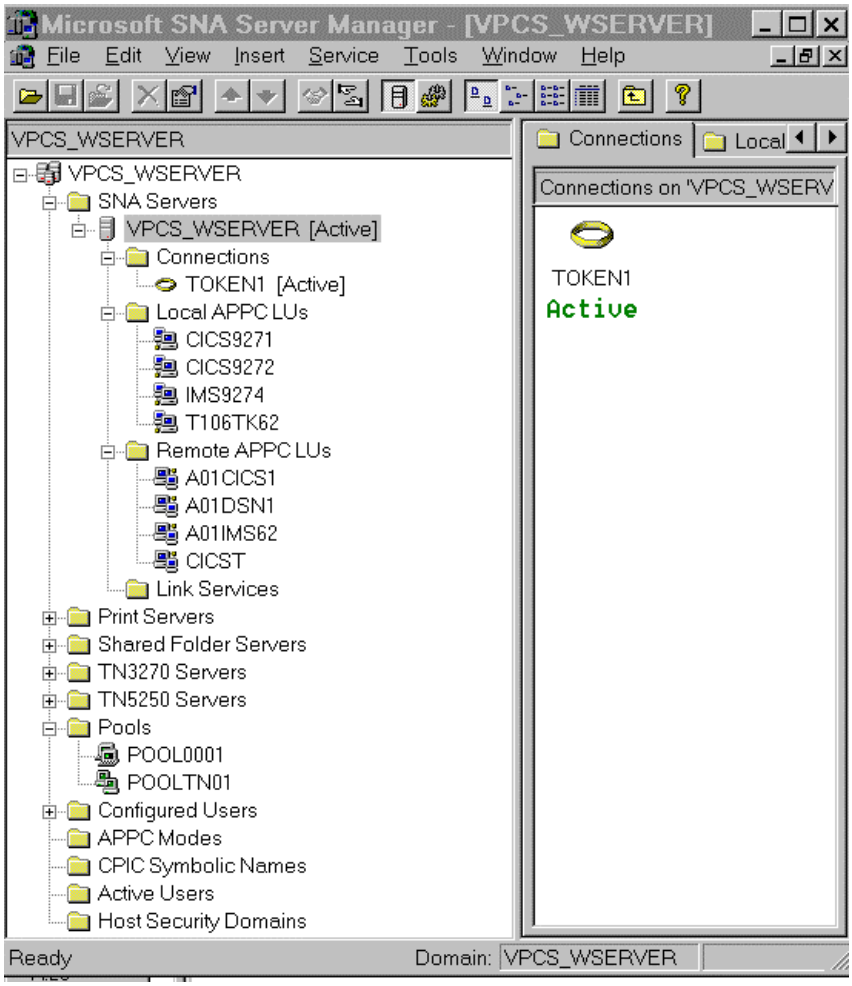


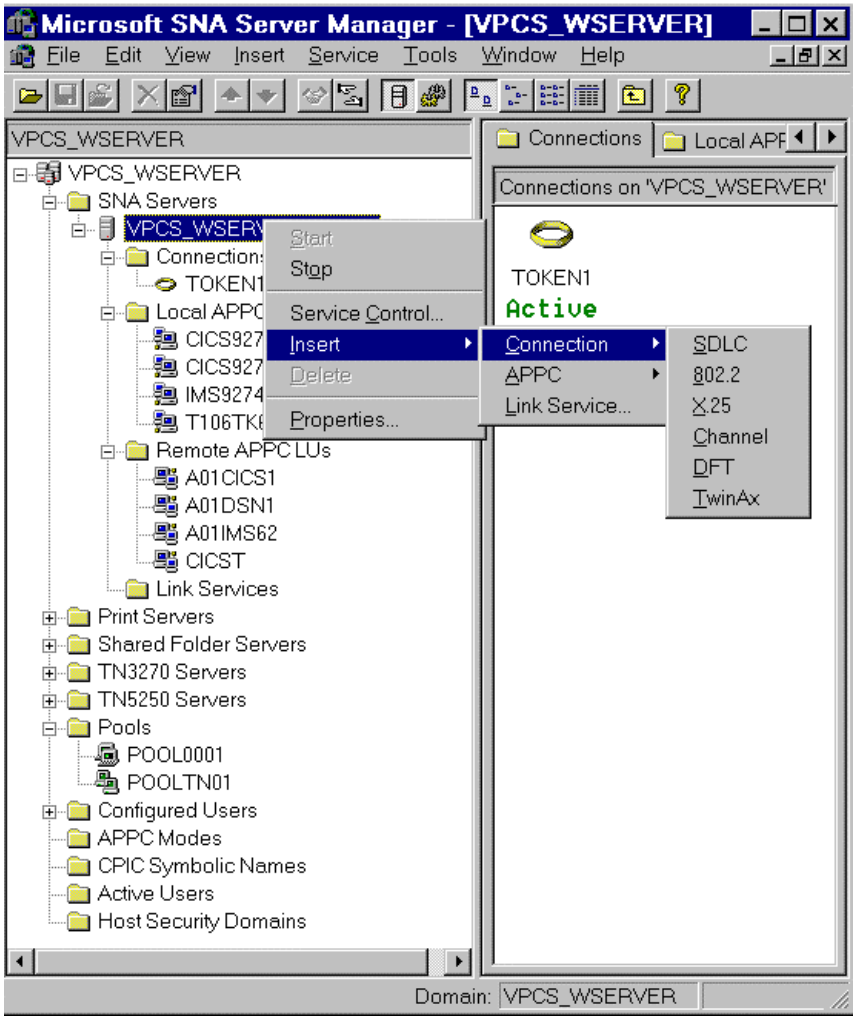
9.2.4. Configuration Windows NT

9.2.4.1. Configuration SNA Server 3.0A

(Le Service Pack 2 doit être appliqué pour des corrections APPC).

Les paramètres SNA Server doivent correspondre aux définitions VTAM, NCP et CICS du Site Central MVS.

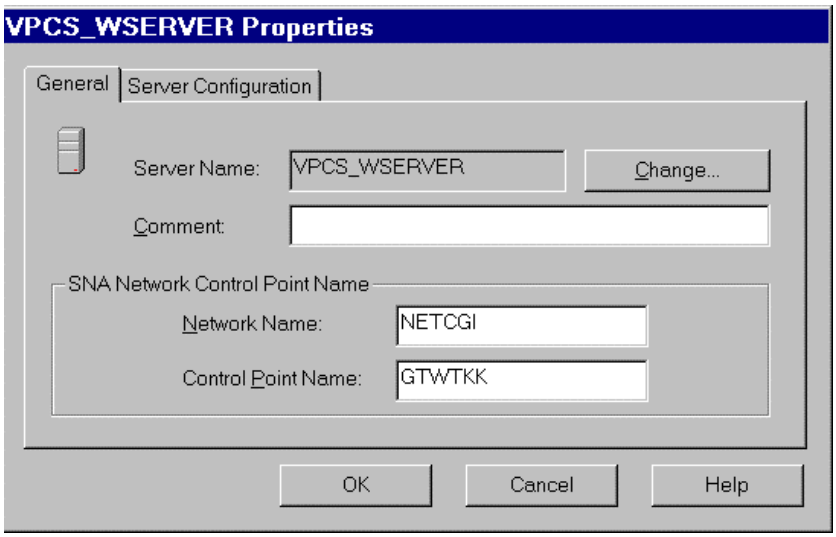




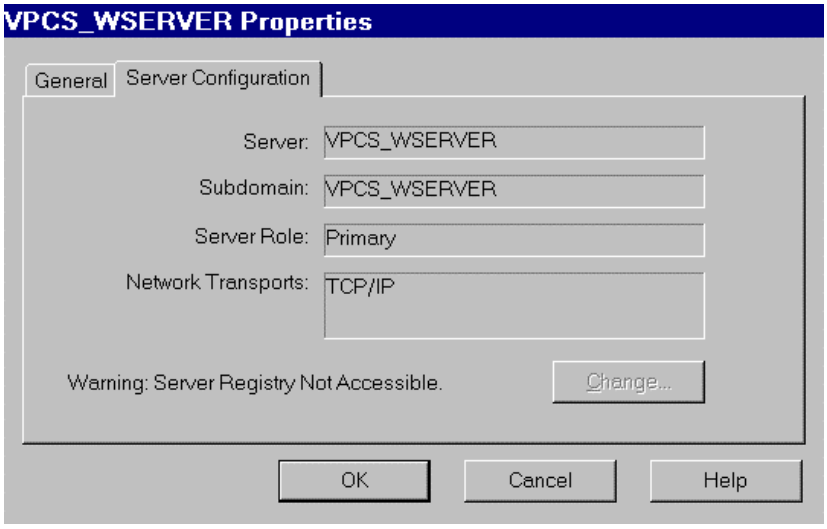
9.2.4.1.1. Propriétés du Serveur

`Network Name` = identifiant du réseau SNA (NETID dans l'ATCSTRxx de VTAM)

`Control Point Name` = correspondant au CPNAME dans la définition de la PU de VTAM



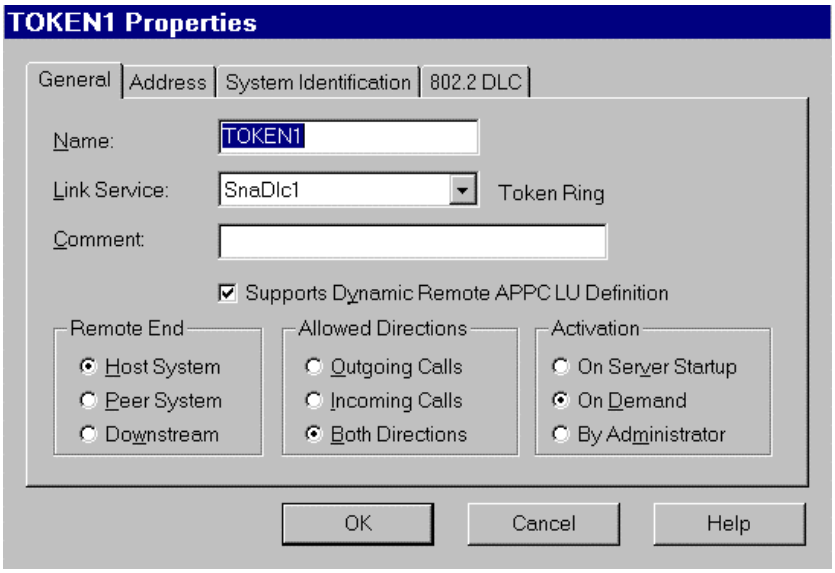
- Type de transport utilisé entre le client et le serveur SNA (ex: protocole TCP/IP)



9.2.4.1.2. Propriétés du Lien

`Link Service` : Le type de lien doit être choisi dès l'installation ou installé ultérieurement par le programme Setup. C'est le composant de SNA Server qui communique avec le driver de la carte réseau. Le SNA DLC 802.2 Link service est alloué pour la communication avec le site central dans un réseau LAN Token ring ou Ethernet.

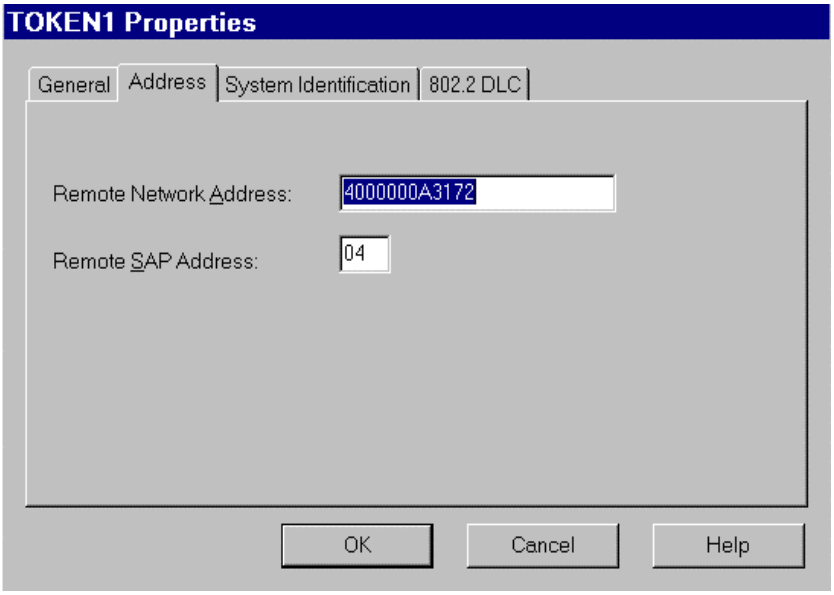
Type de connexion = 802.2



- Définition du TIC contrôleur 3745 dans NCP (attachement au réseau LAN).

```
************************************************
*          TIC BNN                          05742090
************************************************
A01L1TK  LINE ADDRESS=(1088,FULL),
                    PORTADD=01,
                    LOCADD=4000000A3172,
                    ISTATUS=ACTIVE,
                    UACB=(X$P1AX,X$P1AR)
*
A01TK1PU PU ISTATUS=ACTIVE,
                    ADDR=01
************************************************
```

`Remote Network Address` = LOCADD (MAC Address) dans la macro LINE du NCP.



- Paramètres `SSCPNAME` et `NETID` de l'ATCSTRxx de VTAM (nécessaire pour la configuration du Remote Node Name dans SNA Server) :

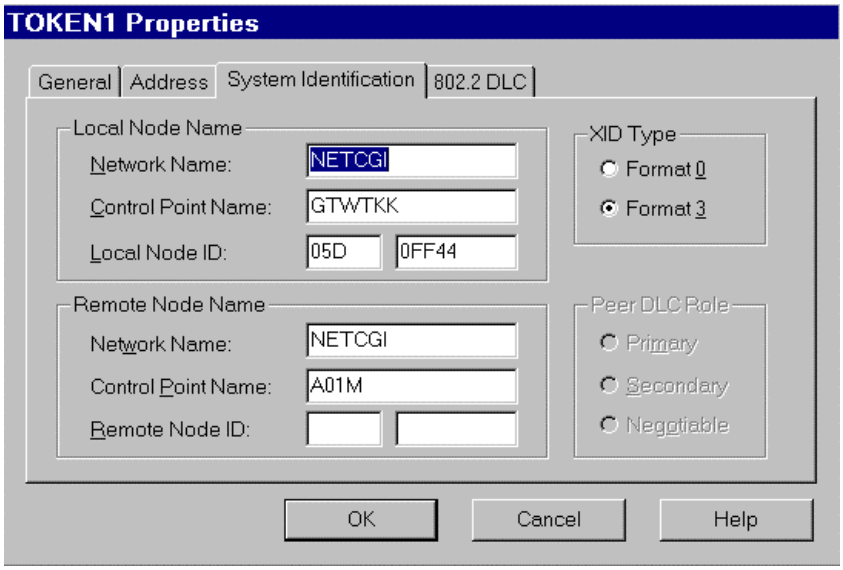
```
*********************************
NOPROMPT,CONFIG=00,SSCPID=01,
MAXSUBA=31,SUPP=NOSUP,
SSCPNAME=A01M,
SSCPORD=DEFINED,
NETID=NETCGI,
HOSTSA=1,
CRPLBUF=(550,,20,,40,40),
IOBUF=(420,182,25,,40,40),
LFBUF=(300,,0,,20,10),
LPBUF=(50,,0,,5,5),
SFBUF=(50,,0,,5,5),
SPBUF=(90,,0,,5,5),
NOTRACE,TYPE=VTAM
*********************************
```

- Définition de la PU dans VTAM correspondant à la passerelle SNA Server (utilisée dans la configuration du Local Node Name pour SNA Server) :

```
*********************************************
*/* SWITCHED MAJOR NODE                      *
*********************************************
*
SW6TKR   VBUILD TYPE=SWNET,MAXNO=12,MAXGRP=06
*
W6TK00   PU    ADDR=55,
                CPNAME=GTWTKK,
```

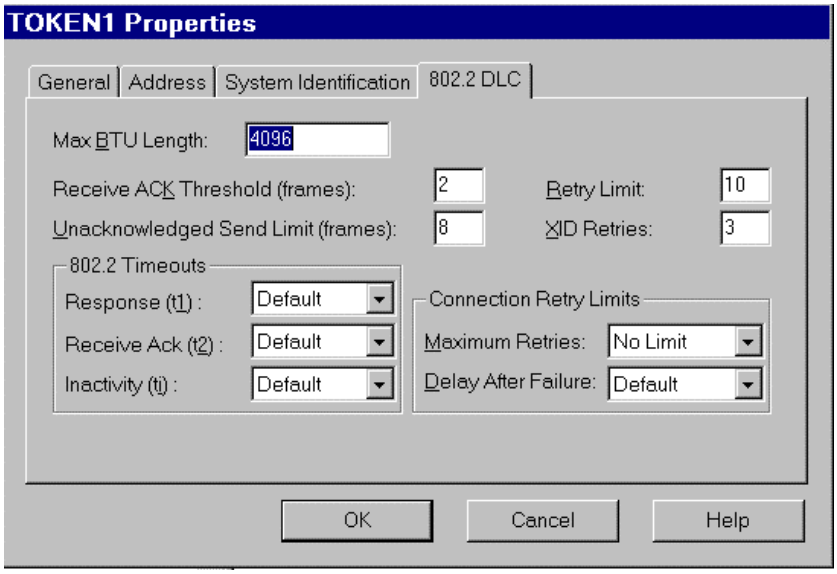
```
                IDBLK=05D,
                IDNUM=0FF44,
                DYNLU=YES,
                MAXPATH=1,
                DISCNT=YES,
                IRETRY=YES,
                VPACING=7,
                PACING=7,
                SSCPFM=USSSCS,
                USSTAB=USSTAB2,
                MAXDATA=4096,
                PUTYPE=2,
                MAXOUT=7,
                DATMODE=FULL
*
*  ==>  INDEPENDENT  LU
*
CICS9271 LU     LOCADDR=0,
                ISTATUS=ACTIVE,
                MODETAB=MTLU62,
                DLOGMOD=LU62
*
*******************************************************
```

- `Local Node Name:`
- `Local Node ID` = IDBLK & IDNUM
- `Control Point Name` = CPNAME
- `Network Name` = NETID (ATCSTRxx)
- `Remote Node Name:`
- `Control Point Name` = SSCPNAME
- `Network Name` = NETID (ATCSTRxx)



`Max BTU Length` (frame size) correspond au MAXDATA de la PU dans VTAM.

- pour adaptateur Token ring à 4 Mbps, doit être inférieur ou égal à 4195
- pour adaptateur Ethernet, doit être inférieur ou égal à 1493.

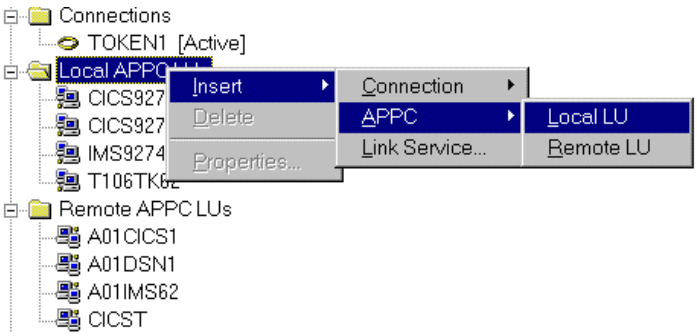


9.2.4.1.3. Local APPC LU

APPC utilise une locale LU (indépendante ou dépendante) et une ou plusieurs LUs éloignées. Les sessions APPC communiquent entre deux LUs (Local et Remote).

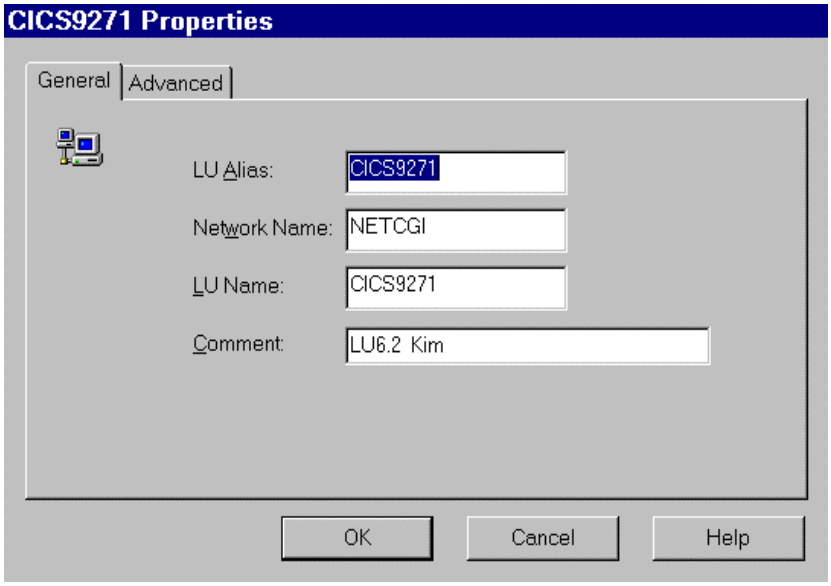
Le minima requis sur MVS pour communiquer avec un TP (Transaction Program) :

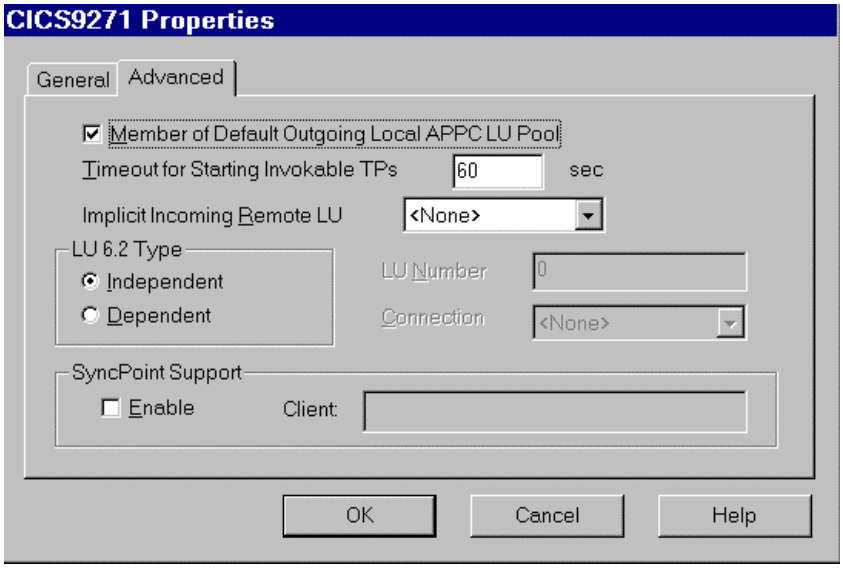
- VTAM version 3.2
- NCP version 5.2 (3745)
- NCP version 4.3 (3725)



Local LU est défini avec la PU dans VTAM comme une LU Indépendante avec LOCADDR = 0:

- `LU Name` = CICS9271
- `Network Name` = NETID dans ATCSTRxx
- `LU Alias` = Identifiant de la LU en local pour TPs (Transaction Programs)





9.2.4.1.4. Remote APPC LU

- Définition de l'APPL CICS dans VTAM :

```
*************************************************************
*/*     THIS MEMBER CONTAINS VTAM APPLICATION DEFINITION
*/*     .     NAME          ACBNAME
*/*           ------------  -------------
*/*     .     A01CICS1      CICST
*************************************************************
A01CICS1 APPL  EAS=160,
                    ACBNAME=CICST,                APPLID FOR ACB
                    AUTH=(ACQ,VPACE,PASS),
                    PARSESS=YES,
                    SONSCIP=YES,
                    MODETAB=MTLU62
*************************************************************
```

- Dans la table SIT du CICS, InterSystem Communication doit être activé : ISC=YES

- Définition de la CONNEXION dans CICS :

  Netname (CICS9271) correspond au Local APPC LU

OVERTYPE TO MODIFY *CICS RELEASE = 0330*
CEDA ALter
Connection : SG71
Group : GRPISC9
DEscription ==> CONNEXION LU6.2 SNA SERVER
CONNECTION IDENTIFIERS
Netname ==> ***CICS9271***
INDsys ==>
REMOTE ATTRIBUTES
REMOTESystem ==>
REMOTEName ==>
CONNECTION PROPERTIES
ACcessmethod ==> Vtam
Protocol ==> Appc
SInglesess ==> No
DAtastream ==> User
RECordformat ==> U
OPERATIONAL PROPERTIES
AUtoconnect ==> Yes
INService ==> Yes
SECURITY
SEcurityname ==> SYTD
ATtachsec ==> Verify
BINDPassword ==>
BINDSecurity ==> No
APPLID=CICST

- Définition de la SESSION dans CICS :
  - `Connection` (SG71) correspond au code connexion defini ci-dessus
  - `MOdename` (LU62) correspond au nom du Mode défini dans SNA Server
  - `MAximum` (004 , 002) correspond aux paramètres Parallel Session Limit et au Partner Min Contention Winner défini dans le Mode (LU62) de SNA Server
  - `SENDSize` et `RECEIVESize` correspondent aux paramètres `Max Receive RU Size` et `Max Send RU Size` du Mode (LU62) de SNA Server

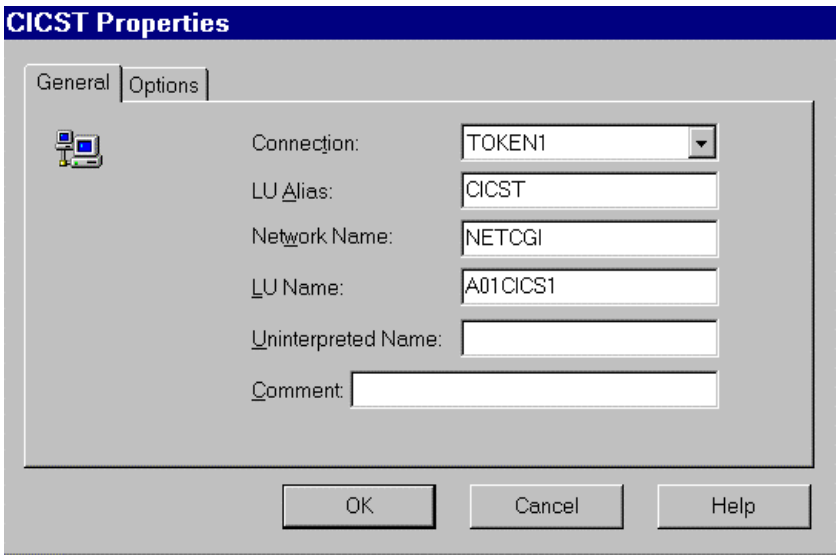
---

OVERTYPE TO MODIFY *CICS RELEASE = 0330*
CEDA ALter
Sessions : SESSIO71
Group : GRPISC9
DEscription ==> SESSION LU6.2 SNA Server
SESSION IDENTIFIERS
Connection ==> ***SG71***
SESSName ==>
NETnameq ==>
MOdename ==> ***LU62***
SESSION PROPERTIES

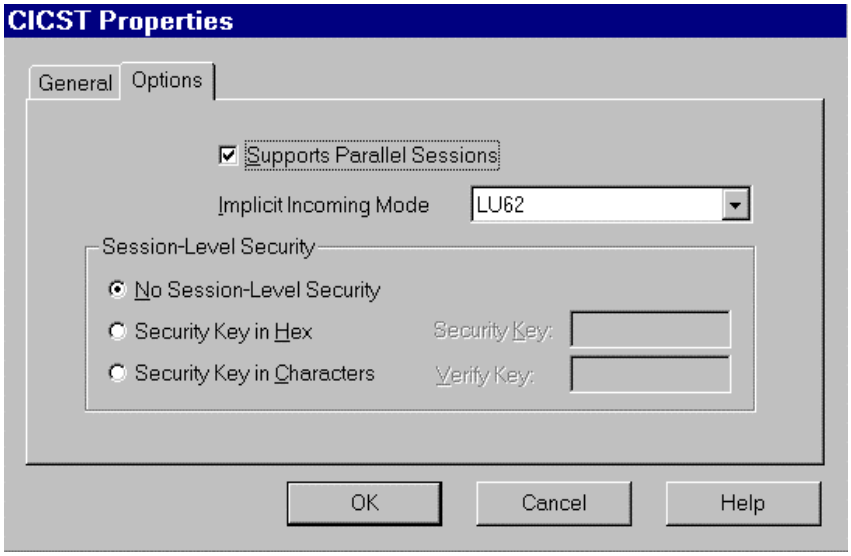
Protocol ==> Appc
*MAximum ==> **004 , 002***
RECEIVEPfx ==>
RECEIVECount ==>
SENDPfx ==>
SENDCount ==>
*SENDSize ==> **08192***
*RECEIVESize ==> **08192***
SESSPriority ==> 000
Transaction :
OPERATOR DEFAULTS
OPERId :
OPERPriority : 000
OPERRsl : 0
OPERSecurity : 1
PRESET SECURITY
USERId ==>
OPERATIONAL PROPERTIES
Autoconnect ==> Yes
INservice :
Buildchain ==> Yes
USERArealen ==> 000
IOarealen ==> 00000 , 00000
RELreq ==> Yes
DIscreq ==> No
NEPclass ==> 000
RECOVERY
RECOVOption ==> Sysdefault
RECOVNotify ==> None

APPLID=CICST

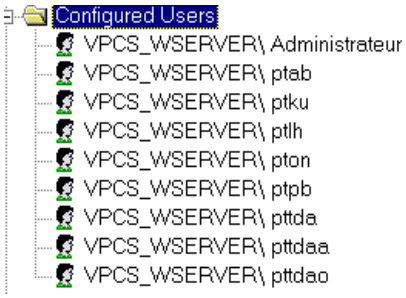
- Définition du Remote APPC LU dans SNA Server:
  - `Network Name` = NETID dans ATCSTRxx
  - `LU Alias` = Identifiant de la LU en local TPs (Transaction Programs)
  - `LU Name` = APPL ID CICS dans VTAM (A01CICS1)

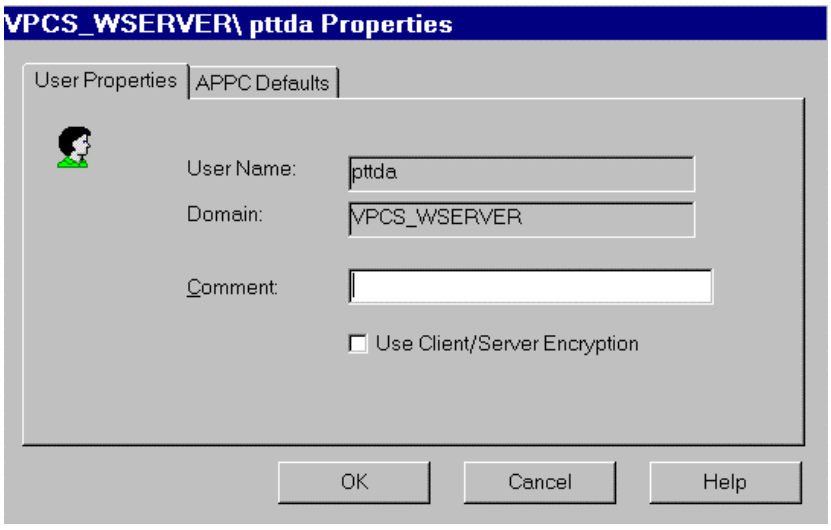


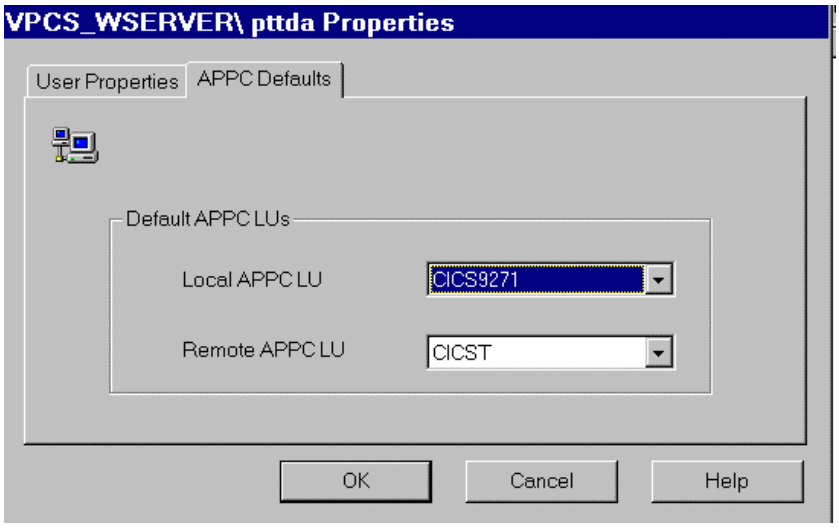
- `Implicit Incoming Mode` = Mode Name utilisé pour Supports Parallel Sessions et défini dans l'APPC Modes de SNA Server



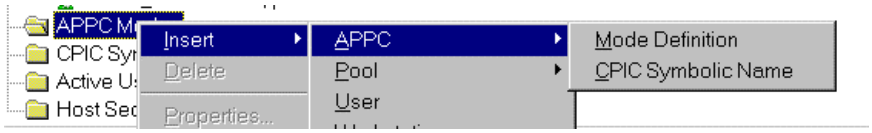
9.2.4.1.5. Configuration des Utilisateurs







9.2.4.1.6. Définition du MODE



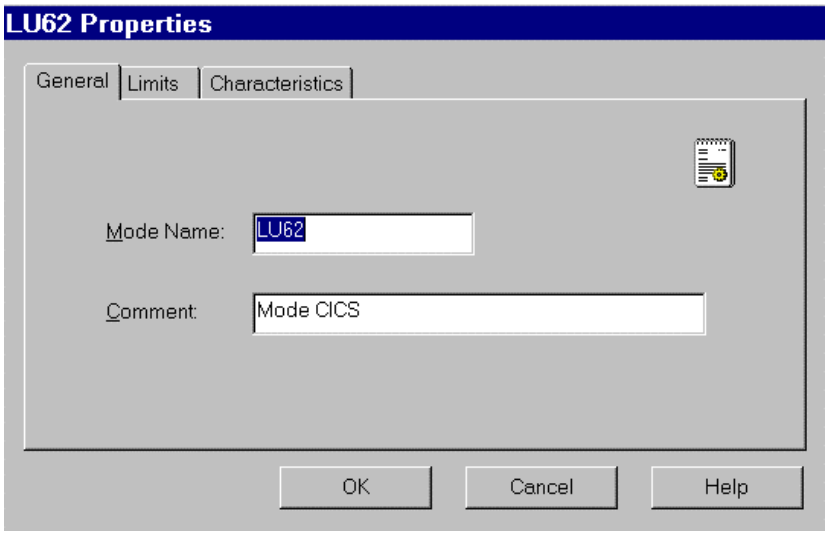
- Définition des MODES LU6.2 dans la table des Modes de VTAM :

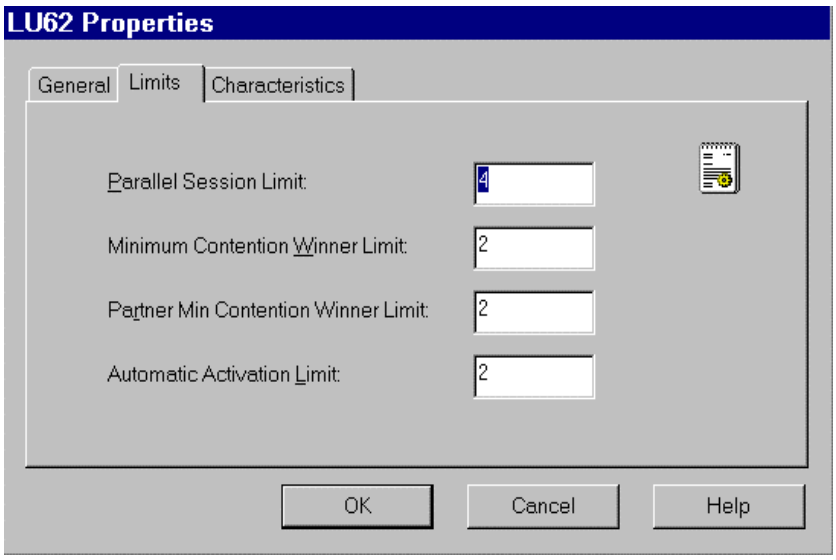
  Le Mode LU62 est un exemple de mode configuré et utilisé avec APPC (LU6.2). Le mode SNASVCMG est inclus dans SNA Server et est utilisé par le "Supports Parallels Sessions".

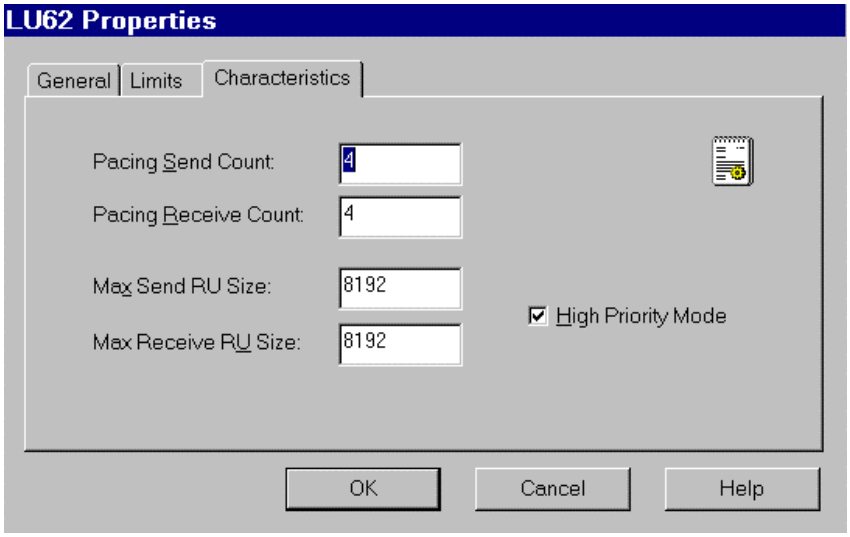
```
--------------------------------------------------------------------------------
*
***     TITLE '--- "MODTABLE" CONCERNANT LES LU 6.2 ---'    *******
*
SNASVCMG MODEENT LOGMODE=SNASVCMG,
                                FMPROF=X'13',
                                TSPROF=X'07',
                                 PRIPROT=X'B0',
                                SECPROT=X'B0',
                                COMPROT=X'D0B1',
                                RUSIZES=X'8585',
                                ENCR=B'0000',
                                PSERVIC=X'060200000000000000000300'
*
LU62     MODEENT LOGMODE=LU62,
                                TYPE=X'00',
                                FMPROF=X'13',
                                TSPROF=X'07',
                                PRIPROT=X'B0',
                                SECPROT=X'B0',
                                COMPROT=X'50B1',
                                RUSIZES=X'8989',
                                SRCVPAC=X'00',
                                PSNDPAC=X'00',
                                SSNDPAC=X'00',
                                PSERVIC=X'06020000000000000002C00'
```

- Définition du Mode dans SNA Server :

  `Mode Name` = Mode défini dans la table des modes de VTAM



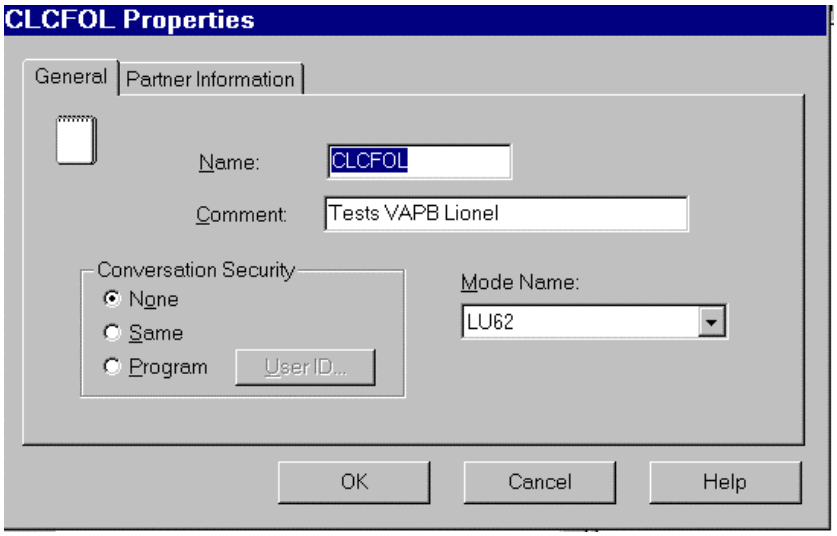




9.2.4.1.7. Définition du CPI-C Symbolic Destinations Names (Side Information)

CPI-C est l'interface API de communication. Il fournit un groupe standard d'appels de fonction pour toutes les plates-formes APPC prenant en charge CPI-C.

Le `Name` correspond au nom externe du moniteur de communication Pacbase (établit la connexion entre les clients et les programmes serveur utilisant APPC).



- Définition de la transaction CICS :

```
---------------------------------------------------------------------------
OVERTYPE TO MODIFY                          CICS RELEASE = 0330
CEDA  ALter
TRansaction   : VP20
Group         : VISUAL
DEscription  ==> VISUAL/CLIENT WIN/NT CPIC
PROGram      ==> CLCFOL
TWasize      ==> 00000
PROFile      ==> DFHCICST
```

PArtitionset ==>
STatus ==> Enabled
PRIMedsize : 00000
TASKDATALoc ==> Below
TASKDATAKey ==> User
REMOTE ATTRIBUTES
DYnamic ==> No
REMOTESystem ==>
REMOTEName ==>
TRProf ==>
Localq ==>
SCHEDULING
PRIOrity ==> 001
TClass ==> No
ALIASES
Alias ==>
TASKReq ==>
XTRanid ==>
TPName ==>
==>
XTPname ==>
==>
==>
RECOVERY
DTimout ==> No
Indoubt ==> Backout
RESTart ==> No
SPurge ==> No
TPUrge ==> No
DUmp ==> Yes
TRACe ==> Yes
SECURITY
RESSec ==> No
Cmdsec ==> No
Extsec : No
TRANsec : 01
RSl : 00

APPLID=CICST

Si un programme utilise DB2 au cours de la transaction, il est nécessaire de rattacher le plan DB2 à la transaction du moniteur serveur en la déclarant dans la table RCT du CICS/ESA :

DSNCRCT TYPE=ENTRY,TXID=(VP20),THRDM=2,
THRDA=2,PLAN=VP20,AUTH=(USERID,*,*)

Si l'application fait des accès DB2, il est nécessaire d'autoriser et de rattacher la transaction utilisateur au plan DB2 dans la table RCT de la région CICS :

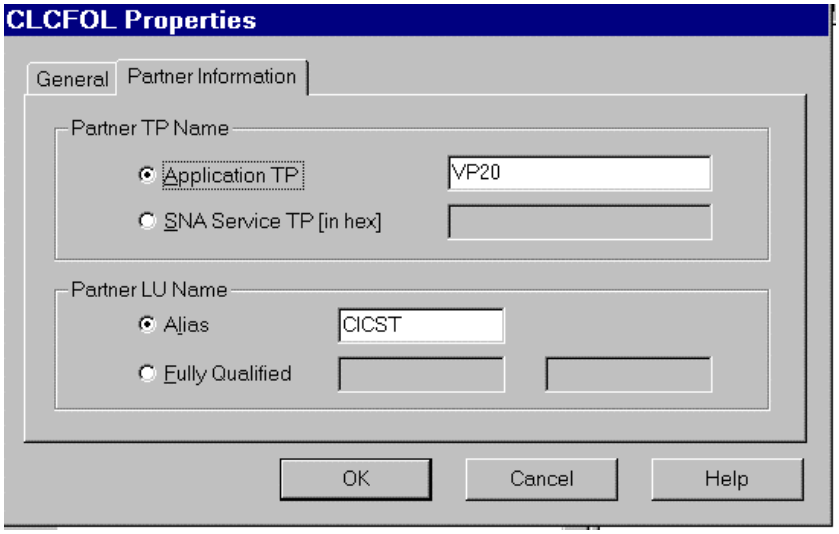
DSNCRCT TYPE=ENTRY,TXID=(**VP20**),THRDM=6,
THRDA=6,PLAN=**VP20**,AUTH=(USERID,*,*)

- Définition du Partner TP Name :

  `Application TP` = Code Transaction activant le moniteur de communication sur le Host et défini ci-dessus

- Définition du Partner LU Name :

  `Alias` = Alias défini dans Remote APPC LU



9.3. CICS ECI

9.3.1. Configuration MVS et CICS

9.3.1.1. Définitions VTAM

Minima requis : VTAM Version 3.3

9.3.1.1.1. Définition de l'ATCSTR de VTAM

```
*********************************
NOPROMPT,CONFIG=00,SSCPID=01,
MAXSUBA=31,SUPP=NOSUP,
SSCPNAME=A01M,
SSCPORD=DEFINED,
NETID=NETCGI,
HOSTSA=1,
CRPLBUF=(550,,20,,40,40),
IOBUF=(420,182,25,,40,40),
LFBUF=(300,,0,,20,10),
LPBUF=(50,,0,,5,5),
SFBUF=(50,,0,,5,5),
SPBUF=(90,,0,,5,5),
NOTRACE,TYPE=VTAM
*********************************
```

9.3.1.1.2. MAC Address du TIC (attachement LAN) contrôleur 3745 dans NCP :

```
************************************************
*          TIC BNN                             05742090
************************************************
A01L1TK  LINE ADDRESS=(1088,FULL),
              PORTADD=01,
              LOCADD=400003172000,
              ISTATUS=ACTIVE,
                        UACB=(X$P1AX,X$P1AR)
*
A01TK1PU PU ISTATUS=ACTIVE,
```

```
                ADDR=01
        ********************************************
```

9.3.1.1.3. Définition du CICS

```
A01CICS1 APPL  EAS=160,                ESTIMATED CONCURRENT SESSIONS   *
               ACBNAME=CICST,          APPLID FOR ACB                  *
               AUTH=(ACQ,VPACE,PASS),  CICS CAN ACQUIRE & PASS TMLS    *
               PARSESS=YES,                   ⇨       Supports parallel Sessions
               SONSCIP=YES,                                           *
               MODETAB=MTLU62            ⇨      nom de la table des modes
```

9.3.1.1.4. Définition du mode

- Définition des caractéristiques pour les Sessions LU6.2
- Le Mode SNASVCMG est utilisé avec le support "Parallels Sessions".

```
TITLE '--- "MODTABLE" CONCERNANT LES LU 6.2 ---'
*
MTLU62   MODETAB
         SPACE 4
SNASVCMG MODEENT LOGMODE=SNASVCMG, FMPROF=X'13',
               TSPROF=X'07', PRIPROT=X'B0', SECPROT=X'B0',
               COMPROT=X'D0B1', RUSIZES=X'8585', ENCR=B'0000',
               PSERVIC=X'060200000000000000000300'
LU62     MODEENT LOGMODE=LU62, TYPE=X'00', FMPROF=X'13',
               TSPROF=X'07', PRIPROT=X'B0', SECPROT=X'B0',
               COMPROT=X'50B1', RUSIZES=X'8787', SRCVPAC=X'00',
               PSNDPAC=X'00', SSNDPAC=X'00',
PSERVIC=X'060200000000000000002C00'
```

9.3.1.1.5. Définition de la passerelle SNA

```
*/* LIB: SYS1.VTAMLST(SW1TKR)
*/*
*/*
*/* SWITCHED MAJOR NODE TOKEN-RING ST-MARC :            -  06/07/95.
*/*   --->    LIEN XCA MAJOR NODE ==> XCA1TKR  (IBM3172-3)
*/*   --->    LIEN GROUPE XCA     ==> GRP02
*              ------------------------------
*    - MODEL FOR IDBLK X'05D' - OS/2 COMMUNICATIONS MANAGER
*              ------------------------------
*
SW1TKR   VBUILD TYPE=SWNET,MAXNO=99,MAXGRP=10
*
*          ----------------------------------------
* -------> DEFINING A GATEWAY TOKEN-RING --> GTWTK1  <--------
*          ----------------------------------------
W1TK00   PU    ADDR=50,
               CPNAME=GTWTK1,
               IDBLK=05D,
               IDNUM=00002,
               DYNLU=YES,
               MAXPATH=1,
               DISCNT=NO,
               IRETRY=YES,
               VPACING=7,
               PACING=7,
```

```
SSCPFM=USSSCS,
MAXDATA=4096,
PUTYPE=2
```

L'option `DYNLU=YES` permet d'éviter toute définition complémentaire de Lu 6.2 au niveau VTAM (pour les postes du réseau TR).

9.3.1.1.6. Définition d'une LU indépendante

Malgré ce qui vient d'être dit ci-dessus, il peut être intéressant de définir une LU indépendante pour les premiers tests de communication :

```
*/* LIB: SYS1.VTAMLST(SW1TKR)
*/*
*/*
*/* SWITCHED MAJOR NODE TOKEN-RING ST-MARC :            -  06/07/95.
*/*
*/*
*  INDEPENDENT LUS
*
CGI5075           LU    LOCADDR=0,
                  ISTATUS=ACTIVE,
                  DLOGMOD=LU62,
                  MODETAB=MTLU62
```

9.3.1.2. Définitions APPC/MVS

Minima requis : MVS/ESA Version 4.2

Il n'y a pas de définitions spécifiques car on utilise la couche APPC livrée avec la version de CICS.

9.3.1.3. Définitions CICS

Minima requis : CICS/ESA Version 4.1

9.3.1.3.1. Paramètre de l'InterSystem Communication dans la table SIT

ISC=YES

9.3.1.3.2. Connexion

```
  Connection     : SGFB
  Group          : GRPISC5
  DEscription    :
 CONNECTION IDENTIFIERS
  Netname        : CGI5075    ⇨      à déclarer comme Local LU dans CM/2 (codification
libre)
  INDsys         :
 REMOTE ATTRIBUTES
  REMOTESystem   :
  REMOTEName     :
 CONNECTION PROPERTIES
  ACcessmethod   : Vtam
  Protocol       : Appc
  SInglesess     : No         ⇨      dans le cas d'une LU indépendante
  DAtastream     : User
  RECordformat   : U
 OPERATIONAL PROPERTIES
  AUtoconnect    : Yes
```

```
  INService      : Yes
 SECURITY
  SEcurityname   : PTPD
ATtachsec      : Verify       ⇨      Verification du userid et password à la conversation
  BINDPassword   :
  BINDSecurity   : No
```

9.3.1.3.3. Session

```
  Sessions       : SESSIOFB
  Group          : GRPISC5
  DEscription    :
 SESSION IDENTIFIERS
  Connection     : SGFB         ⇨      code de la connexion définie ci-dessus
  SESSName       :
  NETnameq       :
  MOdename       : LU62         ⇨      mode défini dans la MODTABLE de VTAM
 SESSION PROPERTIES
  Protocol       : Appc
  MAximum        : 004 , 002
  RECEIVEPfx     :
  RECEIVECount   :
  SENDPfx        :
  SENDCount      :
  SENDSize       : 08192
  RECEIVESize    : 08192
  SESSPriority   : 000
  Transaction    :
 OPERATOR DEFAULTS
  OPERId         :
  OPERPriority   : 000
  OPERRsl        : 0
  OPERSecurity   : 1
 PRESET SECURITY
  USERId         :
 OPERATIONAL PROPERTIES
  Autoconnect    : Yes
  INservice      :
  Buildchain     : Yes
  USERArealen    : 000
  IOarealen      : 00000 , 00000
  RELreq         : Yes
  DIscreq        : No
  NEPclass       : 000
 RECOVERY
  RECOVOption    : Sysdefault
  RECOVNotify    : None
```

9.3.1.3.4. Définition de la transaction

Définition d'une transaction Miroir utilisateur

```
------------------------------------------------------------
                                                CICS RELEASE = 0410
   TRANSaction    : VEC1
   Group          : VPCS250
   DEscription    :
   PROGram        : DFHMIRS
   TWasize        : 00000             0-32767
   PROFile        : DFHCICSA
   PArtitionset   :
   STAtus         : Enabled           Enabled ! Disabled
   PRIMedsize     : 00000             0-65520
   TASKDATALoc    : Below             Below ! Any
   TASKDATAKey    : User              User ! Cics
   STOrageclear   : No                No ! Yes
   RUnaway        : System            System ! 0-2700000
   SHutdown       : Disabled          Disabled ! Enabled
   ISolate        : Yes               Yes ! No
  REMOTE ATTRIBUTES
+  DYnamic        : No                No ! Yes
REMOTESystem   :
 REMOTEName     :
 TRProf         :
 Localq         :                   No ! Yes
SCHEDULING
 PRIOrity       : 001               0-255
 TClass         : No                No ! 1-10
 TRANClass      : DFHTCL00
ALIASES
 Alias          :
 TASKReq        :
 XTRanid        :
 TPName         :
                :
 XTPname        :
                :
                :
RECOVERY
  DTimout        : No                No ! 1-6800
  INdoubt        : Backout           Backout ! Commit ! Wait
  RESTart        : No                No ! Yes
  SPurge         : No                No ! Yes
  TPUrge         : No                No ! Yes
  DUmp           : Yes               Yes ! No
  TRACe          : Yes               Yes ! No
  COnfdata       : No                No ! Yes
 SECURITY
  RESSec         : Yes               No ! Yes
  CMdsec         : Yes               No ! Yes
  Extsec         : No
  TRANSec        : 01                1-64
  RSl            : 00                0-24 ! Public
           ------------------------------------------------------------
```

De plus, si une transaction fait des accès à une base DB2, il faut autoriser et rattacher celle-ci au plan DB2. Le code transaction et le plan DB2 doivent donc être déclaré dans la Table RCT. Par défaut le transaction CPMI (Transaction

Miroir) est utilisée avec le CICS Client. Dans cet exemple le plan DB2 utilisé avec application Cliente se nomme ATDF :

```
DSNRCT TYPE = ENTRY,TXID=(CPMI,CSPM,VICL),
       THRDM=2,THRDA=2,PLAN=ATDF,AUTH=(USERID,*,*)
```

Si la transaction miroir CPMI provoque un Abend ACN1, c'est que la table de conversion DFHCNV n'est pas définie dans la région CICS. Elle est nécessaire pour l'utilisation de cette transaction. Définir cette table puis assembler et LinkEdit avec les paramètres suivants :

```
*********************************************************************************
*******
      //EXBCCNV  JOB (009),'BC',CLASS=X,MSGCLASS=X,NOTIFY=SYTD
      //CICSCNV  EXEC DFHAUPLE,
      // PARM.LNKEDT='RENT,REUS,LIST,XREF,LET,NCAL,AMODE=31,RMODE=ANY'
      //ASM.SYSLIB DD DSN=CICS330.SDFHMAC,DISP=SHR
      //ASSEM.SYSUT1 DD DSN=PT$EXP.CICST330.SOURCE(DFHCNV),DISP=SHR
      //LNKEDT.SYSLMOD DD DSN=PT$PDV.PB80204.MTR8,DISP=SHR

     DFHCNV   TYPE=INITIAL
     DFHCNV   TYPE=ENTRY,RTYPE=PC,RNAME=TESTVP,CLINTCP=(850,437),  *
        SRVERCP=297
     DFHCNV   TYPE=SELECT,OPTION=DEFAULT
     DFHCNV   TYPE=FIELD,OFFSET=0,DATATYP=CHARACTER,DATALEN=8051,  *ES
        LAST=YES
     DFHCNV   TYPE=FINAL
     END

*********************************************************************************
*******
```

9.3.2. Configuration OS/2

9.3.2.1. Communication Manager/2 : configuration APPC

Minima requis : Communication Manager/2 Version 1.11

9.3.2.1.1. Choix de la communication

Référez-vous à l'écran contenu dans le paragraphe équivalent du sous-chapitre 9.1, *IMS CPI-C*.

9.3.2.1.2. Réseau Token Ring

Référez-vous à l'écran contenu dans le paragraphe équivalent du sous-chapitre 9.1, *IMS CPI-C*..

9.3.2.1.3. Token Ring DLC Adapter

Référez-vous à l'écran contenu dans le paragraphe équivalent du sous-chapitre 9.1, *IMS CPI-C*.

9.3.2.1.4. Local Node

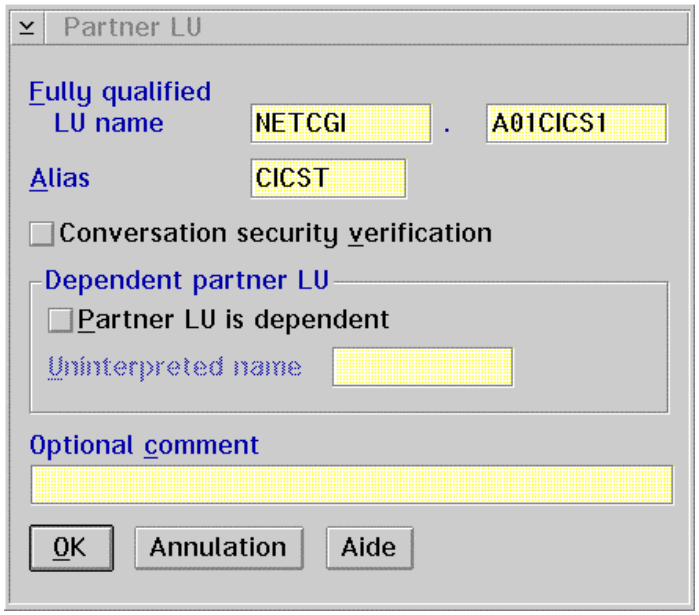
Référez-vous à l'écran contenu dans le paragraphe équivalent du sous-chapitre 9.1, *IMS CPI-C*.

9.3.2.1.5. Connections List

Référez-vous à l'écran contenu dans le paragraphe équivalent du sous-chapitre 9.1, *IMS CPI-C*.

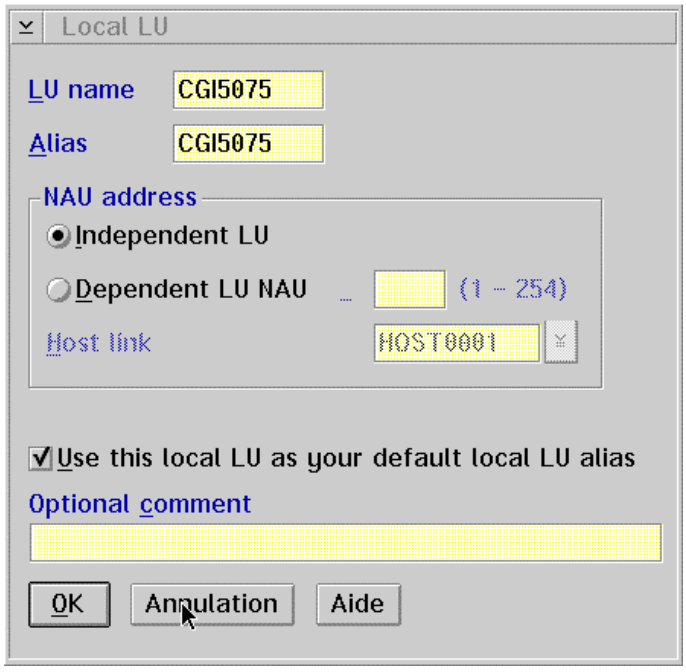
9.3.2.1.6. Partner LU

La LU Partenaire doit être créée à partir du bouton poussoir `Define Partner Lus...` de l'écran précédent.



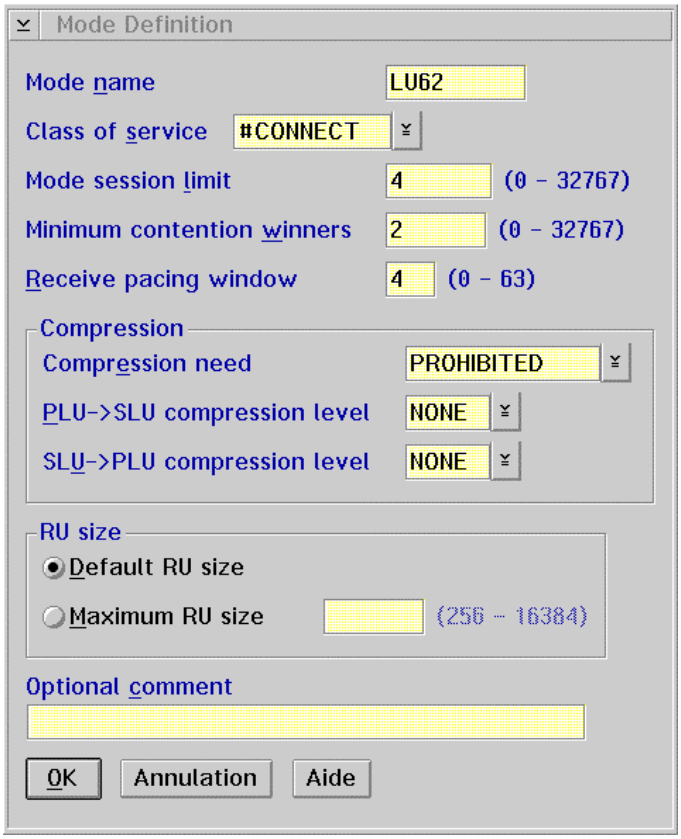
`LU Name` : paramètre `NETID` de la définition générale de VTAM (NETCGI) + paramètre `APPL` de définition du CICS dans la VTAM List (`A01CICS1`).

9.3.2.1.7. Local LU



`LU Name` : Cette Local Lu doit être déclarée au niveau CICS Host dans le Netname de la connexion.

9.3.2.1.8. Mode



`Mode Name` : mode défini dans la table des modes VTAM, paramètre `LOGMODE` de la macro-instruction `MODEENT`

9.3.2.2. CICS OS/2

9.3.2.2.1. CICS Universal Client for OS/2 Version 3.xx

La configuration de CICS OS/2 Client est effectuée directement dans le fichier `CICSCLI.INI` dans le répertoire `\CICSCLI\BIN` (installation standard).

```
;*********************************************************************
;* IBM CICS Client - Initialization File                             *
;*********************************************************************

;---------------------------------------------------------------------
; Client section - This section defines the local CICS client.  There
;                  should only be one Client section.

Client = CGI5075        ; Auto-install client on the server
   MaxServers = 1       ; Only allow one server connection
   MaxRequests = 20           ; Limit the maximum server interaction
   MaxBufferSize = 32         ; Allow for a 32K maximum COMMAREA
   LogFile = CICSCLI.LOG      ; Set the error log file name
   TraceFile = CICSCLI.TRC    ; Set the trace log file name
   DosMemory = 48       ; The DOS client's memory pool size

;---------------------------------------------------------------------
; Server section - This section defines a server to which the client may
;                  connect.  There may be several Server sections.

Server = CICSSNA              ; Arbitrary name for the server
   Description = CICS Server  ; Arbitrary description for the server
   Protocol = SNA             ; Matches with a Driver section below
```

```
   NetName = NETCGI.A01CICS1  ; The server's NetBIOS name
   Adapter = 0                ; Use NetBIOS on LAN adapter 0
   UpperCaseSecurity = N           ; Fold Userid and Password to uppercase
   LocalLUName=CGI5075
   Modename=LU62

Driver = CM2APPC        ; Matches the Server's Protocol value
   DriverName = CCLIBMSN      ; Use the IBM CM/2 APPC
```

9.4. TCP-IP Socket vers UNIX

9.4.1. Configuration

L'API Middleware SOCKET permet la communication entre une application VisualAge et des serveurs COBOL Microfocus situés dans un environnement UNIX par l'intermédiaire d'un listener s'exécutant sur la machine UNIX à atteindre.
Le paramétrage se limite donc aux options de compilation et de Link et aux librairies SQL et Microfocus nécessaires que l'on peut trouver dans l'exemple de makefile qui suit.

9.4.1.1. Exemple de makefile

```
CC=cc
CFLGS= -c  -D_PAC_AIX
CFLGSX= -c  -D_PAC_AIX
COB=cob
COBFLGS=-x  -B static
LIB=/cgi/oracle/procob/lib/cobsqlintf.o \
        /cgi/oracle/lib/libsql.a \
        /cgi/oracle/lib/osntab.o \
        -lora \
        -lsqlnet \
        -lnlsrtl \
        -lcv6 \
        -lcore \
        -lnlsrtl \
        -lcv6 \
        -lcore -lm -lld

LDFLAGS=-L/cgi/oracle/lib

all: serveur

serveur.o: serveur.c serveur.h
      $(CC) $(CFLGS)  serveur.c

serveur: serveur.o
      $(COB) $(COBFLGS) -e "main" $(LDFLAGS) $(LIB)  serveur.o  -o serveur
```

Ce makefile, donné à titre d'exemple, permet de compiler un serveur accédant à une base de données Oracle.

9.4.1.2. Lancement du Listener sur l'UNIX

Le process permettant "l'écoute" du port sur l'UNIX doit être lancé par la commande :

```
serveur path port &
```

avec `path =` chemin permettant l'accès aux DLL des serveurs
`port =` port non utilisé dans le fichier services

9.5. TCP-IP Socket vers Windows/NT

9.5.1. Configuration

L'API Middleware SOCKET permet la communication entre une application VisualAge et des serveurs COBOL Microfocus situés sur une plateforme Windows/NT par l'intermédiaire d'un listener s'exécutant sur la machine Windows/NT à atteindre.
Le seul paramétrage à respecter est donc celui du compilateur Microfocus.

9.5.1.1. Paramétrage de la compilation Microfocus sur Windows/NT

Ce paramétrage est défini dans le fichier `COBOL.DIR` de la directory `\COBOL32\LBR` (pour une installation standard) :

```
VERBOSE
TRACE
VSC2
PERFORM-TYPE"OSVS"
NOBOUND
IBMCOMP
NATIVE "EBCDIC"
ASSIGN "EXTERNAL"
SEQUENTIAL "LINE"
```

La compilation est activée par : `CBLLINK -v -d -s serveur.cbl`

Le paramétrage, donné à titre d'exemple, permet de compiler un serveur de vue logique accédant à des fichiers indexés.
La commande de compilation permet de produire les DLLs des serveurs COBOL.

9.5.1.2. Lancement du Listener sur la machine Windows/NT

Le process permettant "l'écoute" du port doit être lancé par la commande :

```
serveur path port
```

avec `path =` chemin permettant l'accès aux DLL des serveurs
`port =` port non utilisé dans le fichier services

9.6. CICS TCP/IP Sockets Interface

9.6.1. Configuration CICS TCP/IP

9.6.1.1. Prerequisites

MVS/ESA :

- TCP/IP Version 3, Release 1
- CICS/ESA Version 3, Release 3
- CICS TCP/IP Socket Interface Version 3.1

9.6.1.2. CICS Startup

Modification du JCL de lancement pour la région CICS.

```
---------------------------------------------------------------------------------------------------------
//PMTCICST  JOB (008),'CICS TEST PAC',MSGLEVEL=(2,0),CLASS=O,
//   MSGCLASS=X
//CICST    PROC INDEX=CICS330,
//           UTINDX='PT$EXP.CICST330',
//           REGSZE=6M,
//           START='COLD',
//           SIP=T,
//DFHRPL   DD  DSN=&UTINDX..LNK,DISP=SHR
//                  DD  DSN=SYS1.TCPIP310.SEZALINK,DISP=SHR
//                  DD  DSN=TCPIP310.SEZATCP,DISP=SHR
//TCPDATA   DD SYSOUT=&OUTC,DCB=(DSORG=PS,RECFM=V,BLKSIZE=136)
---------------------------------------------------------------------------------------------------------
```

9.6.1.3. Définition transactions CICS TCP/IP

- Listener Task

```
---------------------------------------------------------------
TRansaction    : CSKL
  Group        : TCPIPI
  DEscription    : Listener Task
  PROGram      : EZACIC02
  TWasize      : 00000
  PROFile      : DFHCICST
  PArtitionset   :
  STatus       : Enabled
  PRIMedsize     : 00000
  TASKDATALoc    : Below
  TASKDATAKey    : Cics
 REMOTE ATTRIBUTES
  DYnamic      : No
  REMOTESystem   :
  REMOTEName     :
  TRProf       :
  Localq       :
---------------------------------------------------------------
                  ******************
```

*TCP/IP Version 3.1.0

- Enable the Socket Interface

```
---------------------------------------------------------------------
TRansaction   : CSKE
 Group        : TCPIPI
 DEscription  : Enable Sockets Interface
 PROGram      : EZACIC00
 TWasize      : 00000
 PROFile      : DFHCICST
 PArtitionset :
 STatus       : Enabled
 PRIMedsize   : 00000
 TASKDATALoc  : Below
 TASKDATAKey  : Cics
REMOTE ATTRIBUTES
 DYnamic      : No
 REMOTESystem :
 REMOTEName   :
 TRProf       :
 Localq       :
---------------------------------------------------------------------
```

- Terminate the socket interface

```
---------------------------------------------------------------------
TRansaction   : CSKD
  Group       : TCPIPI
  DEscription : Disable Sockets Interface
  PROGram     : EZACIC00
  TWasize     : 00000
  PROFile     : DFHCICST
  PArtitionset :
  STatus      : Enabled
  PRIMedsize  : 00000
  TASKDATALoc : Below
  TASKDATAKey : Cics
 REMOTE ATTRIBUTES
  DYnamic     : No
  REMOTESystem :
  REMOTEName  :
  TRProf      :
+ Localq      :
---------------------------------------------------------------------
                      ************************
```

*TCP/IP Version 3.2.0

- Configure the socket interface

```
---------------------------------------------------------------------
TRansaction   : EZAC
 Group        : TCPIPI
 DEscription  : CONFIGURE SOCKETS INTERFACE
 PROGram      : EZACIC23
 TWasize      : 00000
 PROFile      : DFHCICST
 PArtitionset :
 STatus       : Enabled
 PRIMedsize   : 00000
 TASKDATALoc  : Below
 TASKDATAKey  : Cics
REMOTE ATTRIBUTES
```

```
 DYnamic       : No
 REMOTESystem  :
 REMOTEName    :
 TRProf        :
 Localq        :
--------------------------------------------------------------------------------
```

- Enable the socket interface

```
--------------------------------------------------------------------
TRansaction   : EZAO
 Group        : TCPIPI
 DEscription  : ENABLE SOCKETS INTERFACE
 PROGram      : EZACIC00
 TWasize      : 00000
 PROFile      : DFHCICST
 PArtitionset :
 STatus       : Enabled
 PRIMedsize   : 00000
 TASKDATALoc  : Below
 TASKDATAKey  : Cics
REMOTE ATTRIBUTES
 DYnamic      : No
 REMOTESystem :
 REMOTEName   :
 TRProf       :
 Localq       :
--------------------------------------------------------------------------------
```

- Terminate the socket interface

```
--------------------------------------------------------------------
TRansaction   : EZAP
  Group       : TCPIPI
  DEscription : DISABLE SOCKETS INTERFACE
  PROGram     : EZACIC22
  TWasize     : 00000
  PROFile     : DFHCICST
  PArtitionset :
  STatus      : Enabled
  PRIMedsize  : 00000
  TASKDATALoc : Below
  TASKDATAKey : Cics
 REMOTE ATTRIBUTES
  DYnamic     : No
  REMOTESystem :
  REMOTEName  :
  TRProf      :
--------------------------------------------------------------------------------
```

9.6.1.4. Définition programmes CICS TCP/IP

```
--------------------------------------------------------------------------------
PROGram      : EZACIC00
   Group     : TCPIPI
   DEscription : Connection Manager
   Language  : Assembler
   RELoad    : No
   RESident  : No
   USAge     : Transient
```

```
    USElpacopy     : No
    Status         : Enabled
    RSl            : 00
    Cedf           : Yes
    DAtalocation   : Any
    EXECKey        : CICS
   REMOTE ATTRIBUTES
    REMOTESystem   :
 +  REMOTEName     :
      Transid      :
  EXECUtionset   : Fullapi
------------------------------------------------------------------------------
------------------------------------------------------------------------------
PROGram        : EZACIC02
   Group         : TCPIPI
   DEscription   : Listener
   Language      : Assembler
   RELoad        : No
   RESident      : Yes
   USAge         : Normal
   USElpacopy    : No
   Status        : Enabled
   RSl           : 00
   Cedf          : Yes
   DAtalocation  : Any
   EXECKey       : CICS
  REMOTE ATTRIBUTES
   REMOTESystem  :
+  REMOTEName    :
Transid        :
  EXECUtionset   : Fullapi
------------------------------------------------------------------------------
------------------------------------------------------------------------------
Mapset         : EZACICM
   Group         : TCPIPI
   Description   : Mapset for Connection Manager
   REsident      : No
   USAge         : Transient
   USElpacopy    : No
   Status        : Enabled
   RSl           : 00
------------------------------------------------------------------------------
------------------------------------------------------------------------------
PROGram        : EZACIC01
    Group         : TCPIPI
    DEscription   : Task Related User Exit
    Language      : Assembler
    RELoad        : No
    RESident      : Yes
    USAge         : Normal
    USElpacopy    : No
    Status        : Enabled
    RSl           : 00
    Cedf          : Yes
    DAtalocation  : Any
    EXECKey       : CICS
   REMOTE ATTRIBUTES
    REMOTESystem  :
 +  REMOTEName    :
Transid        :
```

```
 EXECUtionset   : Fullapi
------------------------------------------------------------------------------
------------------------------------------------------------------------------
PROGram       : EZACIC20
   Group        : TCPIPI
   DEscription    : Initialization/termination for CICS Sockets
   Language       : Assembler
   RELoad         : No
   RESident       : No
   USAge          : Transient
   USElpacopy     : No
   Status         : Enabled
   RSl          : 00
   Cedf           : Yes
   DAtalocation   : Any
   EXECKey        : CICS
  REMOTE ATTRIBUTES
   REMOTESystem   :
 +  REMOTEName     :
Transid       :
 EXECUtionset   : Fullapi
------------------------------------------------------------------------------
------------------------------------------------------------------------------
PROGram       : EZACIC21
   Group        : TCPIPI
   DEscription    : Initialization Module for CICS Sockets
   Language       : Assembler
   RELoad         : No
   RESident       : No
   USAge          : Transient
   USElpacopy     : No
   Status         : Enabled
   RSl          : 00
   Cedf           : Yes
   DAtalocation   : Any
   EXECKey        : CICS
  REMOTE ATTRIBUTES
   REMOTESystem   :
 +  REMOTEName     :
Transid       :
 EXECUtionset   : Fullapi
------------------------------------------------------------------------------
------------------------------------------------------------------------------
PROGram       : EZACIC22
   Group        : TCPIPI
   DEscription    : Termination Module for CICS Sockets
   Language       : Assembler
   RELoad         : No
   RESident       : No
   USAge          : Transient
   USElpacopy     : No
   Status         : Enabled
   RSl          : 00
   Cedf           : Yes
   DAtalocation   : Any
   EXECKey        : CICS
  REMOTE ATTRIBUTES
   REMOTESystem   :
 +  REMOTEName     :
Transid       :
```

```
 EXECUtionset   : Fullapi
------------------------------------------------------------------------------
------------------------------------------------------------------------------
PROGram       : EZACIC23
   Group        : TCPIPI
   DEscription    : Primary Module for Transaction EZAC
   Language       : Assembler
   RELoad         : No
   RESident       : No
   USAge          : Transient
   USElpacopy     : No
   Status         : Enabled
   RSl          : 00
   Cedf           : Yes
   DAtalocation   : Any
   EXECKey        : User
  REMOTE ATTRIBUTES
   REMOTESystem   :
 +  REMOTEName     :
Transid       :
 EXECUtionset   : Fullapi
------------------------------------------------------------------------------
------------------------------------------------------------------------------
PROGram       : EZACIC24
   Group        : TCPIPI
   DEscription    : Message Delivery Module for CICS Sockets
   Language       : Assembler
   RELoad         : No
   RESident       : No
   USAge          : Transient
   USElpacopy     : No
   Status         : Enabled
   RSl          : 00
   Cedf           : Yes
   DAtalocation   : Any
   EXECKey        : CICS
  REMOTE ATTRIBUTES
   REMOTESystem   :
 +  REMOTEName     :
Transid       :
 EXECUtionset   : Fullapi
------------------------------------------------------------------------------
------------------------------------------------------------------------------
PROGram       : EZACIC25
   Group        : TCPIPI
   DEscription    : Cache Module for the Domain Name Server
   Language       : Assembler
   RELoad         : No
   RESident       : No
   USAge          : Normal
   USElpacopy     : No
   Status         : Enabled
   RSl          : 00
   Cedf           : Yes
   DAtalocation   : Any
   EXECKey        : User
  REMOTE ATTRIBUTES
   REMOTESystem   :
 +  REMOTEName     :
Transid       :
```

```
 EXECUtionset   : Fullapi
------------------------------------------------------------------------------
------------------------------------------------------------------------------
PROGram       : EZACICME
   Group        : TCPIPI
   DEscription    : US English Text Delivery Module
   Language       : Assembler
   RELoad         : No
   RESident       : No
   USAge          : Normal
   USElpacopy     : No
   Status         : Enabled
   RSl          : 00
   Cedf           : Yes
   DAtalocation   : Any
   EXECKey        : CICS
  REMOTE ATTRIBUTES
   REMOTESystem   :
 +  REMOTEName     :
Transid       :
 EXECUtionset   : Fullapi
------------------------------------------------------------------------------
```

9.6.1.5. Définition Table DCT

Définition d'une queue data transitoire TCPM pour le LISTENER, dans la table DCT.

```
------------------------------------------------------------------------------
TCPDATA  DFHDCT TYPE=SDSCI,       TCP/IP OUTPUT
         BLKSIZE=136,
         BUFNO=1,
         DSCNAME=TCPDATA,
         RECSIZE=132,
         RECFORM=VARUNBA,
         TYPEFLE=OUTPUT
*
TCPM     DFHDCT TYPE=EXTRA,       USED FOR MESSAGES - SEE
         DESTID=TCPM,                INDDEST=TCPM BELOW
         DSCNAME=TCPDATA
*
TCPIN   DFHDCT TYPE=INTRA,        TCP/IP
         DESTID=TRAA,
         DESTFAC=FILE,
         TRIGLEV=1,
         TRANSID=TRAA
```

9.6.1.6. Définitions et initialisations des fichiers de Configuration (*TCP/IP Version 3.2.0)

EZACONFG : Fichier de configuration Sockets CICS

```
------------------------------------------------------------------------------
File          : EZACONFG
 Group         : SOCKETS
 DEScription    : CICS SOCKETS CONFIGURATION FILE
VSAM PARAMETERS
 DSNAme        : CICS.STM9.SOCKETS.CFG
 Password      :              PASSWORD NOT SPECIFIED
 Lsrpoolid     : 1            1-8 ! None
 DSNSharing    : Allreqs        Allreqs ! Modifyreqs
```

```
 STRings       : 001             1-255
 Nsrgroup      :
REMOTE ATTRIBUTES
 REMOTESystem  :
 REMOTEName    :
 RECORDSize    :                 1-32767
 Keylength     :                 1-255
INITIAL STATUS
 STAtus        : Enabled         Enabled ! Disabled ! Unenabled
Opentime       : Startup         Firstref ! Startup
 DIsposition   : Share           Share ! Old
BUFFERS
 DAtabuffers   : 00002           2-32767
 Indexbuffers  : 00001           1-32767
DATATABLE PARAMETERS
 Table         : No              No ! Cics ! User
 Maxnumrecs    :                 16-16777215
DATA FORMAT
 RECORDFormat  : V               V ! F
OPERATIONS
 Add           : No              No ! Yes
 BRowse        : Yes             No ! Yes
 DELete        : No              No ! Yes
 REAd          : Yes             Yes ! No
 Update        : No              No ! Yes
AUTO JOURNALLING
JOurnal        : No              No ! 1-99
 JNLRead       : None            None ! Updateonly ! Readonly ! All
 JNLSYNCRead   : No              No ! Yes
 JNLUpdate     : No              No ! Yes
 JNLAdd        : None            None ! Before ! AFter ! ALl
 JNLSYNCWrite  : No              Yes ! No
RECOVERY PARAMETERS
 RECOVery      : None            None ! Backoutonly ! All
 Fwdrecovlog   : No              No ! 1-99
 BAckuptype    : Static          Static ! Dynamic
SECURITY
 RESsecnum     : 00              0-24 ! Public
-------------------------------------------------------------------------------------
```

EZACACHE : Fichier qui est nécessaire si vous utilisez la fonction du Cache Domain Name Server

```
-------------------------------------------------------------------------------------------------------
File           : EZACACHE
 Group         : SOCKETS
 DEScription   : DOMAIN NAME SERVER CACHE CONFIGURATION FILE
VSAM PARAMETERS
 DSNAme        : CICS.STM9.SOCKETS.EZACACHE
 Password      :                 PASSWORD NOT SPECIFIED
 Lsrpoolid     : 1               1-8 ! None
 DSNSharing    : Allreqs         Allreqs ! Modifyreqs
 STRings       : 020             1-255
 Nsrgroup      :
REMOTE ATTRIBUTES
 REMOTESystem  :
 REMOTEName    :
 RECORDSize    :                 1-32767
 Keylength     :                 1-255
INITIAL STATUS
 STAtus        : Enabled         Enabled ! Disabled ! Unenabled
Opentime       : Startup         Firstref ! Startup
```

```
 DIsposition    : Old              Share ! Old
BUFFERS
 DAtabuffers    : 00060            2-32767
 Indexbuffers   : 02000            1-32767
DATATABLE PARAMETERS
 Table          : User             No ! Cics ! User
 Maxnumrecs     : 00004000         16-16777215
DATA FORMAT
 RECORDFormat   : V                V ! F
OPERATIONS
 Add            : Yes              No ! Yes
 BRowse         : Yes              No ! Yes
 DELete         : Yes              No ! Yes
 REAd           : Yes              Yes ! No
 Update         : Yes              No ! Yes
AUTO JOURNALLING
JOurnal         : No               No ! 1-99
 JNLRead        : None             None ! Updateonly ! Readonly ! All
 JNLSYNCRead    : No               No ! Yes
 JNLUpdate      : No               No ! Yes
 JNLAdd         : None             None ! Before ! AFter ! ALl
 JNLSYNCWrite   : No               Yes ! No
RECOVERY PARAMETERS
 RECOVery       : None             None ! Backoutonly ! All
 Fwdrecovlog    : No               No ! 1-99
 BAckuptype     : Static           Static ! Dynamic
SECURITY
 RESsecnum      : 00               0-24 ! Public
```

JCL de définition du fichier VSAM **EZACONFG** et configuration de la macro **EZACICD** pour l'environnement Sockets CICS

```
//*************************************************************//
 //*   THE FOLLOWING JOB DEFINES AND THEN LOADS THE VSAM    *//
 //*   FILE USED FOR CICS/TCP CONFIGURATION. THE JOBSTREAM  *//
 //*   CONSISTS OF THE FOLLOWING STEPS.                     *//
 //*    1). DELETE A CONFIGURATION FILE IF ONE EXISTS       *//
 //*    2). DEFINE THE CONFIGURATION FILE TO VSAM           *//
 //*    3). ASSEMBLE THE INITIALIZATION PROGRAM             *//
 //*    4). LINK THE INITIALIZATION PROGRAM                 *//
 //*    5). EXECUTE THE INITIALIZATION PROGRAM TO LOAD THE  *//
 //*        FILE                                            *//
 //*************************************************************//
 //PTCONFIG    JOB  MSGLEVEL=(1,1)
 //*
 //* THIS STEP DELETES AN OLD COPY OF THE FILE
 //* IF ONE IS THERE.
 //*
 //DEL    EXEC  PGM=IDCAMS
 //SYSPRINT DD  SYSOUT=*
 //SYSIN    DD  *
   DELETE -
    CICS.STM9.SOCKETS.CFG -
     PURGE -
     ERASE
 //*
 //*  THIS STEP DEFINES THE NEW FILE
 //*
 //DEFILE EXEC  PGM=IDCAMS
 //SYSPRINT DD SYSOUT=*
```

```
 //SYSIN    DD *
  DEFINE CLUSTER (NAME(CICS.STM9.SOCKETS.CFG) VOLUMES(CICSVOL) -
     CYL(1 1) -
     IMBED -
     RECORDSIZE(150 150) FREESPACE(0 15) -
     INDEXED ) -
     DATA ( -
      NAME(CICS.STM9.SOCKETS.CFG.DATA) -
      KEYS (16 0) ) -
     INDEX ( -
      NAME(CICS.STM9.SOCKETS.CFG.INDEX) )
 /*
 //*
//* THIS STEP ASSEMBLES THE INITIALIZATION PROGRAM
 //*
 //PRGDEF  EXEC PGM=IEV90,PARM='OBJECT,TERM',REGION=1024K
 //SYSLIB    DD DISP=SHR,DSNAME=SYS1.MACLIB
 //          DD DISP=SHR,DSNAME=TCPV32.SEZACMAC
 //SYSUT1    DD UNIT=SYSDA,SPACE=(CYL,(5,1))
 //SYSUT2    DD UNIT=SYSDA,SPACE=(CYL,(2,1))
 //SYSUT3    DD UNIT=SYSDA,SPACE=(CYL,(2,1))
 //SYSPUNCH  DD DISP=SHR,DSNAME=NULLFILE
 //SYSLIN    DD DSNAME=&&OBJSET,DISP=(MOD,PASS),UNIT=SYSDA,
 //          SPACE=(400,(500,50)),
 //          DCB=(RECFM=FB,BLKSIZE=400,LRECL=80)
 //SYSTERM   DD SYSOUT=*
 //SYSPRINT  DD SYSOUT=*
 //SYSIN     DD *
     EZACICD TYPE=INITIAL,        X
          PRGNAME=EZACICDF,       X
          FILNAME=EZACONFG
     EZACICD TYPE=CICS,           X
          TCPADDR=TCPIP,          X
          NTASKS=20,                        X
          DPRTY=10,                 X
          CACHMIN=10,               X
          CACHMAX=20,               X
          CACHRES=5,                X
          ERRORTD=CSKN,             X
          APPLID=A6ECCSM9
     EZACICD TYPE=LISTENER,  X
          TRANID=CSKL,              X
          PORT=9953,                X
           BACKLOG=40,              X
           ACCTIME=30,              X
           GIVTIME=10,              X
           REATIME=300,             X
           NUMSOCK=100,             X
           WLMGN1=CICSSTM9,   X
           MINMSGL=4,               X
           APPLID=A6ECCSM9
     EZACICD TYPE=FINAL
/*
 //*
 //* THIS STEP LINKS THE INITIALIZATION PROGRAM
 //*
 //LINK    EXEC PGM=IEWL,PARM='LIST,MAP,XREF',
 //          REGION=512K,COND=(4,LT)
 //SYSPRINT  DD SYSOUT=*
 //SYSUT1    DD SPACE=(CYL,(5,1)),DISP=(NEW,PASS),UNIT=SYSDA
```

```
//SYSLMOD   DD DSNAME=&&LOADSET,DISP=(MOD,PASS),UNIT=SYSDA,
//           SPACE=(TRK,(1,1,1)),
//           DCB=(DSORG=PO,RECFM=U,BLKSIZE=32760)
//SYSLIN    DD DSNAME=&&OBJSET,DISP=(OLD,DELETE)
//*
//* THIS STEP EXECUTES THE INITIALIZATION PROGRAM
//*
//FILELOAD EXEC PGM=*.LINK.SYSLMOD,COND=(4,LT)
//EZACONFG DD  DSNAME= CICS.STM9.SOCKETS.CFG,DISP=OLD
```

☞ Vous avez aussi la possibilité de configurer l'interface CICS Sockets TCP/IP par la transaction **EZAC.**

JCL de définition du fichier VSAM **EZACACHE** et configuration de la macro **EZACICR** pour utilisation du Cache DNS

```
----------------------------------------------------------------------------------------------------
//*************************************************************************************//
  //*   THE FOLLOWING JOB DEFINES AND THEN LOADS THE VSAM    *//
  //*   FILE USED FOR THE CACHE.  THE DEFINITION CONSISTS OF *//
  //*   TWO IDCAMS STEPS TO PERFORM THE VSAM DEFINITION      *//
  //*   AND A STEP USING EZACICR TO BUILD THE FILE LOAD      *//
  //*   PROGRAM. THE FINAL STEP EXECUTES THE FILE LOAD       *//
  //*   PROGRAM TO CREATE THE FILE.                          *//
  //*************************************************************************************//
  //PTCACHE  JOB  MSGLEVEL=(1,1)
  //*
  //* THIS STEP DELETES AN OLD COPY OF THE FILE
  //* IF ONE IS THERE.
  //*
  //DEL    EXEC  PGM=IDCAMS
  //SYSPRINT DD  SYSOUT=*
  //SYSIN    DD  *
     DELETE -
      CICS.STM9.SOCKETS.EZACACHE -
        PURGE -
        ERASE
  //*
  //*  THIS STEP DEFINES THE NEW FILE
  //*
  //DEFILE EXEC  PGM=IDCAMS
  //SYSPRINT DD SYSOUT=*
  //SYSIN    DD *
    DEFINE CLUSTER (NAME(CICS.STM9.SOCKETS.EZACACHE) VOLUMES(CICSVOL) -
       CYL(1 1) -
       IMBED -
       RECORDSIZE(500 1000) FREESPACE(0 15) -
       INDEXED ) -
       DATA ( -
         NAME(CICS.STM9.SOCKETS.EZACACHE.DATA) -
         KEYS (255 0) ) -
       INDEX ( -
         NAME(CICS.STM9.SOCKETS.EZACACHE.INDEX) )
/*
  //*
  //*  THIS STEP DEFINES THE FILE LOAD PROGRAM
  //*
  //PRGDEF  EXEC PGM=IEV90,PARM='OBJECT,TERM',REGION=1024K
  //SYSLIB    DD DISP=SHR,DSNAME=SYS1.MACLIB
  //           DD DISP=SHR,DSNAME=TCPV32.SEZACMAC
  //SYSUT1    DD UNIT=SYSDA,SPACE=(CYL,(5,1))
```

```
  //SYSUT2    DD UNIT=SYSDA,SPACE=(CYL,(2,1))
  //SYSUT3    DD UNIT=SYSDA,SPACE=(CYL,(2,1))
  //SYSPUNCH  DD DISP=SHR,DSNAME=NULLFILE
  //SYSLIN    DD DSNAME=&&OBJSET,DISP=(MOD,PASS),UNIT=SYSDA,
  //            SPACE=(400,(500,50)),
  //            DCB=(RECFM=FB,BLKSIZE=400,LRECL=80)
  //SYSTERM   DD SYSOUT=*
  //SYSPRINT  DD SYSOUT=*


  //SYSIN     DD *
          EZACICR TYPE=INITIAL
          EZACICR TYPE=RECORD,NAME=ESSONVS1
          EZACICR TYPE=FINAL
/*
  //LINK    EXEC PGM=IEWL,PARM='LIST,MAP,XREF',
  //            REGION=512K,COND=(4,LT)
  //SYSPRINT  DD SYSOUT=*
  //SYSUT1    DD SPACE=(CYL,(5,1)),DISP=(NEW,PASS),UNIT=SYSDA
  //SYSLMOD   DD DSNAME=&&LOADSET(GO),DISP=(MOD,PASS),UNIT=SYSDA,
  //            SPACE=(TRK,(1,1,1)),
  //            DCB=(DSORG=PO,RECFM=U,BLKSIZE=32760)
  //SYSLIN    DD DSNAME=&&OBJSET,DISP=(OLD,DELETE)
  //*
  //*  THIS STEP EXECUTES THE FILE LOAD PROGRAM
  //*
  //LOAD EXEC PGM=*.LINK.SYSLMOD,COND=((4,LT,ASM),(4,LT,LINK))
  //EZACICRF DD DSN= CICS.STM9.SOCKETS.EZACACHE,DISP=OLD
-------------------------------------------------------------------------------------------------------------------
```

9.6.1.7. Définition table PLT (*TCP/IP V320)

Pour démarrage/arrêt automatique de l'interface Sockets CICS (*TCP/IP V320) il est nécessaire d'ajouter le module EZACIC20 dans la table PLT.

Pour le Démarrage automatique, ajoutez en PLTPI après l'entrée DFHDELIM.

```
-----------------------------------------------------------------------------------------
DFHPLT TYPE=ENTRY,PROGRAM=EZACIC20
-----------------------------------------------------------------------------------------
                          ************************
```

Pour l'arrêt automatique ajoutez en PLTSD avant l'entrée DFHDELIM.

```
-----------------------------------------------------------------------------------------
DFHPLT TYPE=ENTRY,PROGRAM=EZACIC20
-----------------------------------------------------------------------------------------
```

9.6.2. Configuration TCP/IP MVS/ESA

9.6.2.1. Modification configuration TCP/IP

Pour l'utilisation des Sockets CICS TCP/IP, il est nécessaire de définir un PORT pour la région CICS dans le fichier de configuration TCP/IP "hlq.PROFILE.TCPIP".

```
;*********************************************
; PROFILE.TCPIP
; =============
;
; ---------------------------------------------------------------------
;
```

```
; NOTES:
;
;   A port that is not reserved in this list can be used by any user.
;   If you have TCP/IP hosts in your network that reserve ports
;   in the range 1-1023 for privileged applications, you should
;   reserve them here to prevent users from using them.
;
;   The port values below are from RFC 1060, "Assigned Numbers."
;

PORT
1415 TCP CSQ9CHIN          ; MQSeries CSQ9
9011 TCP PTMBRUNT          ; TeamConnection
9950 TCP EXCICS9           ; CICS Socket for CICS9
9953 TCP CICSSTM9           ; CICS Socket for A6ECCSM9
-------------------------------------------------------------------------------
```

9.6.2.2. Paramètre TCPJOBNAME dans le fichier *hlq*.TCPIP.DATA

Pour l'initialisation de CICS TCP/IP, il est nécessaire de connaître le nom de la procédure TCP/IP MVS spécifié dans le fichier `hlq.TCPIP.DATA`, paramètre `TCPIPJOBNAME`.

```
;
;*************************************************************
;  Name of Data Set:     TCPIP.DATA                          *
;                                                            *
;  COPYRIGHT = NONE.                                         *
;                                                            *
; This data, TCPIP.DATA, is used to specify configuration    *
;  information required by TCP/IP client programs.           *
;                                                            *
;                                                            *
;  Syntax Rules for the TCPIP.DATA configuration data set:   *
;     treated as a comment.                                  *
;                                                            *
*************************************************************
; TCPIPJOBNAME specifies the name of the started procedure that was
; used to start the TCPIP address space.    TCPIP is the default.
;
TCPIPJOBNAME TCPIP
```

9.6.3. Démarrage et arrêt manuel du CICS TCP/IP

9.6.3.1. Démarrage du CICS TCP/IP

Exécutez la transaction CICS -> **CSKE** pour le démarrage manuel de CICS TCP/IP (*TCP/IP V310).

```
--------------------------------------------------------------------------------
CSKE                                                          EZACIC00
                      CICS TASK-RELATED USER EXIT
                        CONNECTION MANAGER

                      ENABLE CICS-TCP/IP API
```

```
                    TCPIPJOBNAME tcpip--- PORT 9953-
 PF1=HELP        PF3=QUIT                                  EZAME00
--------------------------------------------------------------------------------
                          **************************
```

Exécutez la transaction CICS -> EZAO pour le démarrage manuel de CICS TCP/IP (*TCP/IP V320).

```
--------------------------------------------------------------------------------
EZAO,START,CICS

 APPLID=     ===> A6ECCSM9          APPLID of CICS

--------------------------------------------------------------------------------
                          **************************
```

*Ensuite vérifier par **CEMT I TAS** que la transaction CSKL est active :

```
CEMT I TAS
        STATUS:  RESULTS - OVERTYPE TO MODIFY
        Tas(0000023) Tra(CKAM)         Sus Tas Pri( 255 )
        Tas(0000024) Tra(CKTI)         Sus Tas Pri( 001 )
        Tas(0000027) Tra(DSNC)         Sus Tas Pri( 255 )
        Tas(0000034) Tra(ISER)         Sus Tas Pri( 254 )
        Tas(0000046) Tra(CSKL)         Sus Tas Pri( 255 )
```

9.6.3.2. Arrêt du CICS TCP/IP

Exécutez la transaction CICS => **CSKD** pour l'arrêt manuel de CICS TCP/IP (*TCP/IP V310).

```
--------------------------------------------------------------------------------
CSKD                                                  EZACIC00
        CICS TASK-RELATED USER EXIT
                CONNECTION MANAGER

                DISABLE CICS-TCP/IP API


                ===> 1  QUIESCENT DISABLE-API
                ===> 2  IMMEDIATE DISABLE-API


ENTER ONE OF THE ABOVE DISABLE-API OPTION NUMBERS: 2


PF1=HELP        PF3=QUIT                                  EZAMD00
--------------------------------------------------------------------------------
                     ************************************
```

Exécutez la transaction CICS => **EZAO** pour l'arrêt manuel de CICS TCP/IP (*TCP/IP V320).

```
--------------------------------------------------------------------------------
EZAO,STOP,CICS

 APPLID          ===> A6ECCSM9          APPLID of CICS
 IMMEDIATE       ===>  Y                Enter Yes!No
--------------------------------------------------------------------------------
```

9.6.4. Compilation Cobol

9.6.4.1. Compilation Cobol du programme Moniteur de Communication VisualAge Pacbase.

Modification du JCL de compilation et d'édition des liens pour l'intégration de l'interface CICS Socket TCP/IP.

```
---------------------------------------------------------------------------------------------------
//PTTDASOC  JOB (661),LH,CLASS=X,MSGCLASS=X,MSGLEVEL=(0,0)
//*JCLLIB ORDER=DSNY220.JCLLIB
//*************************************************
//*  Jcl de compil/link pour moniteur de communication VAP *
//*   avec l'interface CICS SOCKET TCP/IP                  *
//************************************************* **
//CBL EXEC DB2SOCK,MEMBER=CLTMVS,LOAD=PT$VIC.VIC.MTR8,
//         DBRMLIB=PT$VIC.VIC.DBRMLIB,SOURCE=PT$VIC.CICS.SOURCE
//LKED.SYSLIB  DD DSN=PT$VIC.TCPIP310.SEZATCP,LIB=SHR
//LKED.SYSIN DD *
   INCLUDE  SYSLIB(EZACICAL)
   INCLUDE  SYSLIB(EZACIC04)
   INCLUDE  SYSLIB(EZACIC05)
   INCLUDE  DB2LOAD(DSNCLI)
   NAME     CLTMVS(R)
 //
```

9.6.5. Définitions CICS pour l'application VisualAge Pacbase

9.6.5.1. Définition du code Transaction

```
---------------------------------------------------------------------------------------------------
TRansaction    : VSO1
  Group         : VISUAL
  DEscription    : VISUAL/CLIENT Transaction SOCKET CICS TCP/IP
  PROGram        : CLTMVS
  TWasize        : 00000
  PROFile        : DFHCICST
  PArtitionset   :
  STatus         : Enabled
  PRIMedsize     : 00000
  TASKDATALoc    : Below
  TASKDATAKey    : User
 REMOTE ATTRIBUTES
  DYnamic        : No
  REMOTESystem   :
  REMOTEName     :
  TRProf         :
+ Localq         :
---------------------------------------------------------------------------
```

9.6.5.2. Définition du programme Moniteur de Communication

```
---------------------------------------------------------------------------
PROGram        : CLTMVS
   Group         : VISUAL
```

```
   DEscription    : Moniteur de communication SOCKET CICS TCP/IP
   Language       : CObol
   RELoad         : No
   RESident       : No
   USAge          : Normal
   USElpacopy     : No
   Status         : Enabled
   RSl            : 00
   Cedf           : Yes
   DAtalocation   : Below
   EXECKey        : User
  REMOTE ATTRIBUTES
   REMOTESystem   :
 +  REMOTEName     :
+  Transid        :
   EXECUtionset   : Fullapi
-------------------------------------------------------------------------------------------
```

9.6.5.3. Définition du fichier de travail VSAM

```
-------------------------------------------------------------------------------------------
File                : FRVABI
   Group            : VISUAL
   DEScription      :
  VSAM PARAMETERS
   DSNAme           : PT$VIC.CICS.FRBIS
   Password         :
   Lsrpoolid        : 1
   DSNSharing       : Allreqs
   STRings          : 001
   Nsrgroup         :
  REMOTE ATTRIBUTES
   REMOTESystem     :
   REMOTEName       :
   RECORDSize       :
   Keylength        :
  INITIAL STATUS
 +  STAtus         : Enabled
-----------------------------------------------------------------------------
```

9.6.6. Configuration Client

9.6.6.1. Paramètres du fichier VAPLOCAT.INI

Il est nécessaire de positionner les variables d'environnement dans le fichier VAPLOCAT.INI pour la communication avec les Sockets CICS.

```
-----------------------------------------------------
<EnvSocketMVS>
LENGTH=08192
MONITOR=CLTMVS              <- Nom du programme moniteur de communication CICS
MWARE=TCPMVS                <- Type de Protocole de Communication pour le Middleware
MWTRANSID=VSO1              <- Code Transaction CICS utilisé pour l'application
MWTIMEOUT=220
MWCODEPAGE=297              <- Code Page du Host pour transcodage ASCII <-> EBCDIC
MWADDRESS=9.134.16.10 9953 <- Adresse TCP/IP MVS et Port Région CICS/ESA
-----------------------------------------------------
```

9.134.16.10 correspond à l'adresse TCP/IP du host MVS/ESA

9951 correspond au port défini pour la région CICS => PMTCICST

9.6.6.2. Middleware VP

La DLL du Middleware VisualAge Pacbase utilisée pour la communication avec les Sockets CICS TCP/IP se nomme => IXOTMVS.DLL.

9.6.6.3. Traces

En cas de problèmes de communication entre votre application Cliente et Serveur, consultez les messages écrits dans la LOG de la région CICS concernée.

Pour vérifier que la transaction a bien été activée par l'application cliente, il est nécessaire sous CICS de passer la commande de display du 1er programme appelé (Moniteur de Communication) avant et après l'exécution de l'application cliente puis ensuite de vérifier que la valeur USE a augmenté de +1, cela signifie que le programme à été exécuté.

CEMT I PROG(CLTMVS)

```
STATUS:  RESULTS - OVERTYPE TO MODIFY
Prog(TESTMQ  ) Len(0084584) Cob Pro Ena Pri     Ced
Res(000) Use(0000001946) Bel Uex Ful
```

9.7. Middleware local

9.7.1. Configuration

L'API Middleware LOCAL permet la communication entre une application VisualAge et des serveurs COBOL Microfocus situés sur le même poste de travail. Le seul paramétrage à respecter est donc celui du compilateur Microfocus.

9.7.1.1. Paramétrage de la compilation Microfocus OS/2

Ce paramétrage est défini dans le fichier `COBOL.DIR` de la directory `\COBOL32\LBR` (pour une installation standard) :

```
VERBOSE
TRACE
VSC2
PERFORM-TYPE "OSVS"
NOBOUND
IBMCOMP
NATIVE "EBCDIC"
ASSIGN "EXTERNAL"
SEQUENTIAL "LINE"
SQL "DB2"
SQLINIT
SQLDB "PTATDF"                    ⇨      Nom du plan DB2
SQLBIND ""
```

La compilation est activée par : `CBLLINK -v -d -s serveur.cbl`

Le paramétrage, donné à titre d'exemple, permet de compiler un serveur de vue logique accédant à une base de données DB2/2.

La commande de compilation permet de produire les DLLs des serveurs COBOL.
Après l'installation du compilateur Microfocus, il faut construire les DLL `SQLINIT.DLL`, `_SQLE3X.DLL` et `_SQLPRLD.DLL`.

9.8. TUXEDO

9.8.1. Client

Tuxedo /WS version 6.x

La seule configuration à mettre en oeuvre consiste à indiquer l'adresse du serveur et le port associé à l'application par l'intermédiaire de la variable d'environnement WSNADDR.

WSNADDR doit respecter la syntaxe suivante :

WSNADDR=0X0002ppppaaaaaaaa
| | ↳ adresse IP sur huit chiffres en hexa
| ↳ N° de port sur quatre chiffres en hexa
↳ domaine AF_INET.

exemple : *WSNADDR=0X00020BB8C0060A5D*
| ↳ 192.6.10.93
↳ 3000

Si la version de Tuxedo /WS le permet (version 6.4 ou ultérieure), cette adresse peut être spécifiée sous la forme suivante :

WSNADDR=//serveur:port
| ↳ N° de port
↳Nom logique ou adresse IP du serveur

exemple : *WSNADDR=//9.143.96.178 :3005*

9.8.2. Serveur

Se référer au manuel TUXEDO

Le seul impératif à respecter est que les noms des services dans un serveur Tuxedo doivent correspondre aux noms externes des programmes.

9.9. MQSERIES

9.9.1. CICS Adapter

Exemple d'une configuration simple d'une application CICS déclenchée par le Trigger Monitor MQSeries.

- Définition REPLY QUEUE

```
Queue name  . . . . . . . . . . VAP.REQUEST.Q
Description . . . . . . . . . : REQUEST QUEUE

Put enabled . . . . . . . . . : Y  Y=Yes,N=No
Get enabled . . . . . . . . . : Y  Y=Yes,N=No
Usage . . . . . . . . . . . . : N  N=Normal,X=XmitQ
```

```
Storage class . . . . . : DEFAULT
Creation method . . . . . : PREDEFINED
Output use count  . . . . . : 0
Input use count . . . . . : 0
Current queue depth . . . . : 0
Default persistence . . . . : Y  Y=Yes,N=No
Default priority  . . . . . : 5  0 - 9
Message delivery sequence . : F  P=Priority,F=FIFO
Permit shared access  . . . : Y  Y=Yes,N=No
Default share option  . . . : S  E=Exclusive,S=Shared
Index type  . . . . . : N  N=None,M=MsgId,C=CorrelId
Maximum queue depth . . . . : 100000     0 - 999999999
Maximum message length  . . : 4194304  0 - 4194304
Retention interval  . . . . : 999999999  0 - 999999999 hours
Creation date . . . . . : 1999-06-23
Creation time . . . . . : 12.59.47

Trigger Definition

Trigger type  . . . . . : E  F=First,E=Every,D=Depth,N=None

   Trigger set  . . . . . : Y  Y=Yes,N=No
   Trigger message priority : 0  0 - 9
   Trigger depth  . . . . . : 1          1 - 999999999
   Trigger data . . . . . :

   Process name . . . . . : VAP.PROCESS.DEF
   Initiation queue . . . . : CICS.INITQ
```

- Définition REPLY QUEUE

```
Queue name  . . . . . . . . . . VAP.REPLY.Q
Description . . . . . . . . . : output QUEUE


Put enabled . . . . . . . . . : Y  Y=Yes,N=No
Get enabled . . . . . . . . . : Y  Y=Yes,N=No
Usage . . . . . . . . . . . . : N  N=Normal,X=XmitQ
Storage class . . . . . . . . : DEFAULT
Creation method . . . . . . . : PREDEFINED
Output use count  . . . . . . : 0
Input use count . . . . . . . : 0
Current queue depth . . . . . : 0
Default persistence . . . . . : Y  Y=Yes,N=No
Default priority  . . . . . . : 0  0 - 9
Message delivery sequence . . : F  P=Priority,F=FIFO
Permit shared access  . . . . : Y  Y=Yes,N=No
Default share option  . . . . : S  E=Exclusive,S=Shared
Index type  . . . . . . . . . : N  N=None,M=MsgId,C=CorrelId
Maximum queue depth . . . . . : 999999999  0 - 999999999
Maximum message length  . . . : 4194304  0 - 4194304
Retention interval  . . . . . : 999999999  0 - 999999999 hours
Creation date . . . . . . . . : 1999-06-23
Creation time . . . . . . . . : 13.00.10

Trigger type  . . . . . . . . : N  F=First,E=Every,D=Depth,N=None

   Trigger set  . . . . . . . : N  Y=Yes,N=No
   Trigger message priority . : 0  0 - 9
   Trigger depth  . . . . . . : 1          1 - 999999999
   Trigger data . . . . . . . :

   Process name . . . . . . . :
   Initiation queue . . . . . :
```

- Définition INITIATION QUEUE

```
Queue name  . . . . . . . . . . CICS.INITQ
Description . . . . . . . . . : CKTI initiation queue

Put enabled . . . . . . . . . : Y  Y=Yes,N=No
Get enabled . . . . . . . . . : Y  Y=Yes,N=No
Usage . . . . . . . . . . . . : N  N=Normal,X=XmitQ
Storage class . . . . . . . . : SYSTEM
Creation method . . . . . . . : PREDEFINED
Output use count  . . . . . . : 0
Input use count . . . . . . . : 1
Current queue depth . . . . . : 0
Default persistence . . . . . : Y  Y=Yes,N=No
Default priority  . . . . . . : 5  0 - 9
Message delivery sequence . . : F  P=Priority,F=FIFO
Permit shared access  . . . . : Y  Y=Yes,N=No
Default share option  . . . . : E  E=Exclusive,S=Shared
Index type  . . . . . . . . . : N  N=None,M=MsgId,C=CorrelId
Maximum queue depth . . . . . : 100        0 - 999999999
Maximum message length  . . . : 4194304    0 - 4194304
Retention interval  . . . . . : 999999999  0 - 999999999 hours
Creation date . . . . . . . . : 1999-05-07
Creation time . . . . . . . . : 18.30.18
Trigger Definition

Trigger type  . . . . . . . . : N  F=First,E=Every,D=Depth,N=None

   Trigger set  . . . . . . . : N  Y=Yes,N=No
   Trigger message priority . : 0  0 - 9
   Trigger depth  . . . . . . : 1          1 - 999999999
   Trigger data . . . . . . . :

   Process name . . . . . . . :
   Initiation queue . . . . . :
    Event Control
```

- Définition PROCESS

```
Process name . . . . . . . . . VAP.PROCESS.DEF
Description  . . . . . . . . . : VAP Process

Application type . . . . . . . : CICS
Application ID . . . . . . . . : AMQM  ← Transaction application Serveur
User data  . . . . . . . . . . : VAP.REQUEST.Q QMGR
```

9.10. CPIC-C/XCP2

9.10.1. Configuration CPI-C/XCP2 – TDS

9.10.1.1. Prerequisites

- GCOS7 ==> ***V7 - TS7458***
- CNP7 ou DATANET ==> **PID ISO/DSA**
- GCOS7 ou GCOS8 ==> **CPI-C/XCP2**
- DPX20 ou ESCALA ==> **Stack OSI et CPI-C/OSI**

9.10.1.2. Configuration Frontal CNP7

- Définition Controleur Ethernet attaché au réseau LAN et de la station ISO

  Le CNP7 doit avoir l'option `ISOPLG` pour les connexions ISO/DSA (option non standard qui demande une license séparée).

```
&************************************************
&*     LOCAL MICROFEP GENERATION      *
&************************************************
&
SC CNP7 LOC  -ADDR 001:003 -BRK AT -ISOPLG
TS CNP7 LOC  -ADDR 001:003
EX CNP7 -MEM 70 -NBCREQ 50
&
&************************************************
&*    LOCAL NETWORK CONTROLLERS       *
&************************************************
SC STMB RMT  -NAT DSA -ADDR 001:001  -SR SR00
SR SR00 ISO  -TS TS00
TS TS00 DIWS -NR NR00 -TPDU 512
NR NR00 LAN1 -PL PL00
PL PL00 CSM1 -ETHAD 0800385F0100 -CB CB01
PL PL01 CSM2 -CB CB01
CB CB01 LAN1 -PL RLN1
PL RLN1 CSMA -CT RLN1 -ETHAD 0800385F0200
CT RLN1 RLNA -PHAD 1
& ********************************************************
& *   Définition attachement réseau LAN Ethernet       *
& ********************************************************
CB CB02 LAN1 -PL RLN2
PL RLN2 CSMA -ETHAD 0800385F0092 -CT RLN2
CT RLN2 RLNA -PHAD 2
&********************************************************
&*  Définition STID STM4 (Escala) sur LAN              *
&********************************************************
SC STM4 RMT  -NAT ISO -ADDR 001:086    -SR SR10
SR SR10 ISO  -TS TS10
TS TS10 DIWS -NR NR10 -TPDU 512
NR NR10 LAN1 -PL PL10
PL PL10 CSM1 -ETHAD 080038210FF8 -CB CB02
&---------------------------------------------------------------
```

9.10.1.3. Configuration Réseau GCOS7 (directive NETGEN)

- Définition du Noeud distant (directive NETWORK)

```
NET STMB QM=1 QFBLKSZ=512 QMBLKSZ=512 SIMU=2;
COMM '-------------------------------------------------';
```

```
COMM '--    S Y S T E M E      L O C A L              --';
COMM '-----------------------------------------------------';
SYS STMB PF=LSYS SCID=001:001 ISL=(5F-01-00,EA01,CBL_MAIN) OBJLIST;
COMM '-----------------------------------------------------';
COMM '--    S Y S T E M E S - R E L A I S / V O I S I N S -';
COMM '-----------------------------------------------------';
SYS CNP7 PF='CNP7/CNS7/A2' SCID=001:003 ISL=(5F-02-00)  OBJLIST;
DEF CT SET WATCH=3000  ;
COMM '-----------------------------------------------------';
COMM '--    S Y S T E M E S    A    D I S T A N C E    --';
COMM '-----------------------------------------------------';
SYS STM4 PF='STID/ISO/SID4'    SCID=001:086 PT=CNP7  OBJLIST ;
COMM '-----------------------------------------------------';
```

- Définition Workstation TDS et correspondants XCP2 (directive DIRECTORY)

```
DIR ;
COMM '*************************************************************************'
COMM '**  APPLICATION TDS1 CPI-C/XCP2 VISUALAGE PACBASE      **
COMM '*************************************************************************'
TDSWKS NAME=TDS1  TMSESS=28  XCP2WKS=WKS1 ;
XCP2WKS NAME=WKS1 MBX=XKS1   MAXSC=16
MAXTX=10, MAXC=10, MAXPOOL=10, MAXPCS=10, MAXCONV=8
SYNCPT=0, SCBUFSZ=(32512,32512), CONVBUFSZ=64512
CONV_ACCEPT=1, CONVCK=1, CONV_USERID=MANDATORY  ;
XCP2COR NAME=LUTXCP2  XCP2WKS=WKS1  ;
XCP2COR NAME=LUJAS2   SC=STM4   MBX=SESJAS2  PARALLEL=1  ;
XCP2POOL  NAME=POOLTDS1  XCP2COR=LUTXCP2  XCP2WKS=WKS1    MAXSC=4 ;
XCP2POOL  NAME=MODEXCP2  XCP2COR=LUJAS2   XCP2WKS=WKS1    MAXSC=4
WINSRCE=2     WINTRGT=2       WINAUTO=2
DRSRCE=0      DRTRGT=1                    ;
EDIR;
```

9.10.1.4. Configuration GCOS7

- Configuration Site Catalog

Modification du projet rattaché à l'application TDS utilisant les services CPI-C/XCP2.
Sous MNCAT (Maintain_Catalog) :

- ajoutez dans la liste des applications la *mailbox de la WorkStation XCP2* (MBX=XKS1) définie dans la directive XCP2WKS de la configuration Réseau Gcos7.
- ajoutez dans la liste des stations, la station ISO correspondant à l'Escala (STM4)
- ajoutez dans la liste des Utilisateurs et sans Mot de Passe :
  - l'Administrateur XCP2 avec le nom générique `tdsname_ADM` (`tdsname` étant le nom de l'application TDS).
  - le correspondant XCP2 distant (XCP2COR NAME=LUJAS2 SC=STM4)

```
---------------------------------------------------------------------------------------------
  PROJECT              -MODIFDATE- ------------JOBCLASS----------------
  ->PT                12.09.97     OB:BDFLT QC:IOFCL KB KC AB AC
                                   AD AE AF JB JD A  B  C  D  E  F
                                   G  H  I  J  K  L  M  N  O
```

```
                              DFLTOUTC :
    ATTRIBUTES
          STD  NMAIN NSTATION NRMS
          mstupb=SITE ostupb=PROJECT mstupi=SITE ostupi=PROJECT
    mass storage volumes
          STM000    STM010    STM090    STM100
    magnetic tape volumes
          *
    application                    -tdscode-
    ->IOF          DFLT
    ->TDS1                         0FFFFFFF
    ->XKS1                         0FFFFFFF
    station
    ->MAIN
    ->STMB         DFLT
    ->STM4
    user               -modifdate-
    ->LUJAS2        DFLT  12.09.97
    ->PTTDA         DFLT  12.29.97
    ->TDS1_ADM      DFLT  12.09.97
   billing            -modifdate- credit - charge - balance -
   ->613          DFLT  09.26.88  99999999 0        99999999
-------------------------------------------------------------------------------------------
```

9.10.1.5. Configuration TDS

- Préparation de l'environnement TDS

Pour utiliser le protocole XCP2 avec TDS il est nécessaire d'allouer le fichier `PPCLOG` avec l'utilitaire `TP7PREP` :

- `XCP2=YES`
- `XCP2SZ=size`
- `XCP2MD=media`
  (`DEAL=N` pour conserver les fichiers existants*)*

```
-------------------------------------------------------------------------------------------------------------------
15 OVL  HOLD;
16 VL   PRY='SYSFILE=CAT,FILESTAT=CAT,CATNAME=PT,IMPORT=NO',
17      PREV5=MTPREP,PREV6=TP7PREP,
18      PRN='SYSFILE=RSD,FILESTAT=UNCAT',
19      FF='TDS1,MS/D500,STM130,MS/D500,STM130,DEAL=N',
20              GG='DBGSZ=1,MAXDBG=3,CBLSZ=1,SMSZ=15,MAXSM=20,XCP2=YES,NBSW=2,
       SW2SZ=4',
21      VLVL='VL=('&FF','&PRY'';
22 IV   &PREV6 SYS.HSLLIB &VLVL,&GG);
23 SEND '===>  PREPARATION OF 'TDS1' SUCCESSFUL  <===';
-------------------------------------------------------------------------------------------------------------------
```

- Génération TDS

Pour l'utilisation de CPI-C/XCP2 avec TDS le membre STDS de la `tdsname`. SLLIB doit être adapté puis une nouvelle génération TDS (TP7GEN) doit être effectuée.

- TDS Section

Cette clause définit le délai d'attente pour les appels CPI-C (cette directive doit être insérée avant la clause `USE PROCEDURE`) :
`MAXIMUM XCP2-WAITTIME`

```
---------------------------------------------------------------------------------
TDS SECTION.
PROGRAM-ID. TDS1.
BTNS            IS BTNS.
NUMBER OF DUMMY CORRESPONDENT IS 4 MAXIMUM IS 8.
SIMULTANEITY         5.
RESERVE             16 AREAS.
NUMBER MODULES       10.
MESSAGE-LENGTH       6000.
TPR-TIME-LIMIT       45000.
MAXIMUM OF XCP2-WAITTIME IS 300.
USE "MENU"    TRANSACTION-MENU.
USE ZAR100.
USE ZAR200.
---------------------------------------------------------------------------------
```

- Transaction Section

La définition de la transaction utilisant CPI-C/XCP2 dans la clause `MESSAGE-ID` doit comporter la sub-clause :
`XCP2 SERVICE USED`
Pour ouvrir des sessions CPI-C/XCP2 avec le TDS, il est nécessaire d'avoir défini au préalable une transaction avec cette clause.

```
-----------------------------------------------------
 MESSAGE "VIC2" ASSIGN    CLCFOL
    IMPLICIT COMMITMENT
    XCP2 SERVICE USED
    PAGES              50
    WITH TPR ACCOUNTING
    AUTHORITY-CODES     31
    TRANSACTION-STORAGE  SIZE  500.
-----------------------------------------------------
```

- Gestion des POOLs et des CORRESPONDANTs CPI-C/XCP2
  Les commandes seront exécutées sur le Master TDS

- Ouverture du Pool de sessions XCP2

La commande `OPEN_COR_POOL` (abreviation `OCPOOL`) permet d'ouvrir un ou plusieurs pools de sessions reliant une application TDS locale et une application partenaire.

`OCPOOL LUJAS2 MODEXCP2 TDS=TDS1`

```
-------------------------------------------------------------------------------------
 1/1                OPEN_COR_POOL                    -->:

                open correspondent pool

COR              + correspondent name              LUJAS2
POOL             pool name (or *) (xcp2-cor only)  MODEXCP2
ATTRIBUTE          address extension (xcp1-cor only)
ACTIVE_SE          active session number (xcp1-cor only)
TDS              tds name                          TDS1
```

- Liste du Pool de sessions XCP2 :

```
LSCPOOL LUJAS2 MODEXCP2 TDS=TDS1
```

```
---------------------------------------------------------------------------------------
1/1                LIST_COR_POOL                    -->:

                list correspondent pool

COR             + correspondent name             LUJAS2
POOL            pool name (or *) (xcp2-cor only)         MODEXCP2
ATTRIBUTE         address extension (or *) (xcp1-cor only)
NETGEN          known to netgen?                 0
PRINT_MEMBER      print member name
TDS             ds name                          TDS1
---------------------------------------------------------------------------------------
```

- Fermeture du Pool de sessions XCP2 :

La commande `CLOSE_COR_POOL` (abreviation `CLCPOOL`) permet de fermer un ou plusieurs pools de sessions reliant une application TDS locale et une application partenaire.

```
CLCPOOL LUJAS2 MODEXCP2 TDS=TDS1
```

```
---------------------------------------------------------------------------------------
 1/1                CLOSE_COR_POOL                   -->:

                close correspondent pool

 COR             + correspondent name             LUJAS2
 POOL            pool name (or *) (xcp2-cor only)       MODEXCP2
 ATTRIBUTE         address extension (or *) (xcp1-cor only)
 STRONG          abnormal termination?            0
 DRAIN_SOURCE      drain-source (xcp2-cor only)
 DRAIN_TARGET      drain-target (xcp2-cor only)
 TDS             tds name                         TDS1
---------------------------------------------------------------------------------------
```

- Tests et Debugging des transactions utilisant CPI-C/XCP2

Pour le debugging, lancez une trace sur la transaction. Les informations sont envoyées dans un membre **(**`NomTX____Userid_____01`**)** de la Tdsname.DEBUG.

Exemple d'activation d'une trace pour la transaction VIC2 :

```
SEND_TDS 'TRACE PRINT TX=VIC2 USER=*' TDS=TDS1
```

9.10.2. Configuration CPI-C/OSI sur Serveur AIX

S'assurer que les services de communications stack OSI sont installés et chargés sur le serveur AIX. Ils sont nécessaires pour les services CPI-C/OSI.

Dans le Menu "Gestion système"

- Select -> Applications de communication et services

-> OSI Networking

-> OSI Configuration

-> Load the last selected configuration

9.10.2.1. Définition du profile utilisateur XCP2 sous AIX (nécessaire pour configurer CPI- C/XCP2)

 # Profile user XCP2

```
DACUAB=/usr/xcp2/up
DACBIN=/usr/xcp2/ti
DACRSC=/usr/xcp2/conf
DACMEN=/usr/xcp2/men
DACCFG=/usr/xcp2/conf/conf_confcgi
DACSID=/usr/xcp2/side/s2side_cgi
DACSES=/usr/xcp2/ses/s2session_confcgi
DACTRC=1
DACLOG=/usr/xcp2/log
DACTRD=/usr/xcp2/log
export DACBIN DACUAB DACRSC DACCFG DACSES
export DACSID DACMEN DACLOG DACTRD DACTRC
PATH=/usr/bin:/etc:/usr/sbin:/usr/ucb:$HOME/bin:/usr/bin/X11:/sbin:$DACUAB:.
export PATH
```

9.10.2.2. Configuration CPI-C/OSI

La configuration CPI-C/OSI doit correspondre aux définitions de la WorkStation TDS (XCP2WKS) dans la configuration Réseau Gcos7.

Local LU = Correspondant XCP2 (XCP2COR NAME=LUJAS2)

Remote LU = Correspondant XCP2 (XCP2COR NAME=LUTXCP2)

Mode = Pool XCP2 (XCP2POOL NAME=MODEXCP2)

- smit xcp2
- configuration menu
- define resource menu

- Définition NODE

- define Node Server
- add

* add Local Node fields in	**NODEJAS**
* Memory Size (between 50 and 10 000 Kilo bytes)	[**256**]
* Node Operator Messages	**1**
* Logging Messages	**1**
* Verb Message Buffer Size	**512**
* Node Message Buffer Size	**512**

ENTER pour valider puis F3 pour sortir

****

- Définition LOCAL LU

- ***define Local LU***
- *add*

* Add Local LU fields in	**LUJAS2**
* Logical Unit ID (between 1 to 254)	[**2**]
* Network Qualifier	[**NONE**]

* Network Name [**NONE**]
* Default Timeout (-1 to 3600 seconds or NONE) [**-1]**
* Maximum Transaction Programs (1 to 255) [**25**]

ENTER pour valider puis F3 pour sortir

****

- Définition REMOTE LU

- ***define Remote LU***
- *add*

* Add Remote LU fields in **LUTXCP2**
* Network Qualifier [**NONE**]
* Network Name [**NONE**]
* Parallel Sessions **1**
* LU-LU Password in hexadecimal [**NONE**]
* Security Acceptance **NONE**
* Logical Unit ID (between 1 to 254, or NONE) [**5**]

ENTER pour valider puis F3 pour sortir

****

- Définition du MODE

- ***define Mode***
- *add*

* Add Mode fields in **MODEXCP2**
* Minimum RU Length (8 to 256) [**256**]
* Maximum RU Length (256 to 4096) [**4092**]
* Session Reinit **OPER**
* Maximum Sessions (1 to 255) [**4**]
* Minimum First Speaker Sessions (0 to 255) [**2**]
* Auto-initiated Sessions (0 to 255) [**2**]
* Allow Queued Binds **1**

ENTER pour valider puis F3 pour sortir

****

- Définition PARTNER LU

- ***define Partner***
- *add*

* Add Partner fields in **PARTTDS1**
* Remote LU Name **LUTXCP2**
* Mode Name **MODEXCP2**

ENTER pour valider puis F3 pour sortir

****

- Définition TP

- ***define Transaction Program***
- *add*

* Add Transaction Program fields in	**TPALL**
* Program Name	[*****]
* Network Name	[**NONE**]
* Status	**ENABLE**
* Conversation Type	**EITHER**
* Sync Level	**CNFRM**
* Assign LUW	**0**

ENTER pour valider

* Add Transaction Program fields in	**TPCNOS**
* Program Name	[**/usr/xcp2/ti/dac_cnos**]
* Network Name	[**06F1**]
* Status	**ENABLE**
* Conversation Type	**EITHER**
* Sync Level	**CNFRM**
* Assign LUW	**0**

ENTER pour valider puis F3 pour sortir

****

- Définition Configuration

- ***define Configuration***
- *add a Configuration*

* Enter Configuration Name	[**CONFCGI**]

- ENTER pour valider puis F3 pour sortir

- *Set / Change the definitions of an existing Configuration*

- *Select Node Server, ENTER, puis placez le curseur sur CONFCGI*

* Editing configuration	**CONFCGI**
* Select the Configuration's Node Server Name	

- *Ensuite F4 pour lister, puis placez le curseur sur NODEJAS*

* Editing configuration	**CONFCGI**
* Select the Configuration's Node Server Name	**NODEJAS**

- *ENTER pour valider, puis F3 pour sortir*

- *Select Local LU(s), ENTER*
- *Add Resources Definitions, ENTER, puis placez le curseur sur CONFCGI*

* Editing configuration	**CONFCGI**
* Select Local LU Name	

- *Ensuite F4 pour lister, puis placez le curseur sur LUJAS2*

* Editing configuration	**CONFCGI**
* Select Local LU Name	**LUJAS2**

- *ENTER pour valider, puis F3 pour sortir*

- *Select Partner(s), ENTER*

- Add Resources Definitions, ENTER, puis placez le curseur sur CONFCGI

*** Enter Local LU to edit +**

- Ensuite F4 pour lister, puis placez le curseur sur LUJAS2

* Editing configuration **CONFCGI**
* Local LU to edit **LUJAS2**
* Select Partner(s) to Add in the Configuration

- Ensuite F4 pour lister, puis placez le curseur sur PARTTDS1

* Editing configuration **CONFCGI**
* Local LU to edit **LUJAS2**
* Select Partner(s) to Add in the Configuration **PARTTDS1**

- ENTER pour valider, puis F3 pour sortir

- Select Transaction Program(s), ENTER
- Add Resources Definitions, ENTER, puis placez le curseur sur CONFCGI
** Enter Local LU to edit*

- Ensuite F4 pour lister, puis placez le curseur sur LUJAS2

* Editing configuration **CONFCGI**
* Local LU to edit **LUJAS2**
* Select TP(s) to Add in the Configuration

- Ensuite F4 pour lister, puis placez le curseur sur TPALL

* Editing configuration **CONFCGI**
* Local LU to edit **LUJAS2**
* Select TP(s) to Add in the Configuration **TPALL**

- ENTER pour valider

- Ensuite F4 pour lister, puis placez le curseur sur TPCNOS

* Editing configuration **CONFCGI**
* Local LU to edit **LUJAS2**
* Select TP(s) to Add in the Configuration **TPCNOS**

- ENTER pour valider, puis F3 pour sortir

****

- Display Configuration

- Revenir sur Menu "Configuration Menu"
- Select -> Display Configuration

*** Choose Configuration**

- Ensuite F4 pour lister, puis placez le curseur sur CONFCGI

- ENTER pour lister la configuration CONFCGI

LU Name = LUJAS2 NetName = NONE Wait = -1

Partner	Rmt LU	Mode
PARTTDS1	LUTXCP2	MODEXCP2

Program = TPALL Network Name = NONE
Program = TPCNOS Network Name = 06F1

Press the keys <s> and <Enter> to STOP the display of the Configuration
or just <Enter> to CONTINUE...

- ENTER, puis F3 pour sortir

****

** **Génération Configuration***

- Revenir sur Menu "Configuration Menu"
- Select -> Generate Configuration

*** Choose Configuration**

- Ensuite F4 pour lister, placez le curseur sur CONFCGI puis ENTER

Configuration 'CONFCGI' Generated.

- puis F3 pour sortir

****

- Actualisation Fichier Configuration

- Revenir sur Menu "Configuration Menu"
- Select -> Set Current Configuration

* **Choose Configuration**

- Ensuite F4 pour lister, placez le curseur sur CONFCGI puis ENTER

Current Configuration File is CONFCGI.Z.

- puis F3 pour sortir

****

- Configuration ISO

***** La configuration ISO fait référence aux definitions de la WorkStation TDS (XCP2WKS) dans la configuration Réseau Gcos7 et la configuration CNP7.*

** **Définition Local Site LU***
***** Session Selector = Mailbox (MBX) du XCP2COR LUJAS2*
***** Transport Selector = Nom DSA du Noeud AIX (STM4)*
***** Network Selector = Adresse Ethernet AIX*
***** Network = Type de Réseau (ETH pour Ethernet)*

- Revenir sur Menu "Configuration Menu"
- Select -> ISO Session Management
-> Define Configuration
-> Define Local Site LUs
-> Add

*** Enter the local site LU name []**

- Ensuite F4 pour lister, placez le curseur sur LUJAS2, puis ENTER

* Add Local Site Lu **LUJAS2**
* Session Selector [**SESJAS2**]
* Transport Selector [**STM4**]
* Network Selector [**080038210FF8**]
  TRANSPORT Parameters
* Preferred class **4**
* Alternative class **4**
* Flow control **1**
* Credit (1 to 15) [**3**]
* TPDU size [**512**]
* Checksum **0**
* Network **ETH**
  BE CAREFUL :
  The environment variable DACSES (in the profile file) must contain the PATH of a FILE in which is written the PATH of the SESSION FILE.

- ENTER, puis F3 pour sortir

****

- Définition Remote Site LU

***** Session Selector = Mailbox de la WorkStation TDS (MBX=XKS1)*
***** Transport Selector = Nom DSA du Noeud GCOS7 (STMB)*
***** Network Selector = Adresse Ethernet Controleur CNP7*
***** Network = Type de Réseau (ETH pour Ethernet)*

-> Define Remote Site LUs
-> Add

*** Enter the Remote site LU name []**

- Ensuite F4 pour lister, placez le curseur sur LUTXCP2, puis ENTER

* Add Remote Site Lu **LUTXCP2**
* Session Selector [**XKS1**]
* Transport Selector [**STMB**]
* Network Selector [**0800385F0092**]
  TRANSPORT Parameters
* Preferred class **4**
* Alternative class **4**
* Flow control **1**
* Credit (1 to 15) [**3**]
* TPDU size [**512**]
* Checksum **0**
  Network **ETH**
  BE CAREFUL :
  The environment variable DACSES (in the profile file) must contain the PATH of a FILE in which is written the PATH of the SESSION FILE.

- ENTER, puis F3 pour sortir

****

- Activation Trace ISO

-> ISO Session Trace Options

Enter ISO Session trace options:
** Trace Level for ISO Session Interface Calls* ***2***
** Trace Filename ("stdout" for screen)* *[**trace_xcp2**]*

(Trace file will be created in $DACLOG directory.)
BE CAREFUL :
The environment variable DACSES (in the profile file) must contain the PATH of a FILE in which is written the PATH of the SESSION FILE.

- ENTER, puis F3 pour sortir

****

- Display Configuration ISO

- Revenir sur Menu "ISO Session Management"
- Select -> Display Configuration

ISO SESSION FILE : /usr/xcp2/ses/session_confcgi.

LU_name	S_Selector	T_Selector	N_Selector	N_Type
LUJAS2 (loc)	SESJAS2	STM4	080038210FF8	ETH
LUTXCP2 (rmt)	XKS1	STMB	0800385F0092	ETH

- ENTER, puis F3 pour sortir

****

- Fichier Configuration courant

*- Revenir sur Menu "**CPI-C OSI Services**"*
- Select -> Network Operations Menu
-> Set Current Configuration

** Choose Configuration []*

- Ensuite F4 pour lister, placez le curseur sur CONFCGI, puis ENTER

Current Configuration File is CONFCGI.Z.

- puis F3 pour sortir
****

- Activation du NODE

*- Revenir sur Menu "**CPI-C OSI Services**"*
- Select -> Network Operations Menu
-> Activate Node Server

** Really do this* ***1***

- ENTER,

Node Server Activated.

- puis F3 pour sortir

- Select -> Display XCP2 Daemons

PID 46960 Node Server: dac_schd
PID 39538 Node Server: dac_tpi

- F3 pour sortir

****

- Initialisation des Sessions CPI-C/XCP2

*- Revenir sur Menu "**CPI-C OSI Services**"*
- Select -> CPI-C OSI Session Operations Menu
-> Initialize Service Manager Mode

** Local LU Name*
** Remote LU Name*

- Ensuite F4 pour lister, puis placez le curseur sur LUJAS2
* Local LU Name **LUJAS2**
* Remote LU Name

- Ensuite F4 pour lister, puis placez le curseur sur LUTXCP2
* Local LU Name **LUJAS2**
* Remote LU Name **LUTXCP2**

- ENTER pour valider

Mode Initialization Complete.

- puis F3 pour sortir

- Select -> Initialize Session Limits

* Local LU Name
* Remote LU Name
* Mode Name
SESSION Limits
* Maximum Sessions **[]**
* Minimum First Speaker Sessions **[]**
* Minimum Bidder Sessions **[]**

- Ensuite F4 pour lister "Local LU Name", placez le curseur sur LUJAS2 puis ENTER
- Ensuite Select "Remote LU Name", F4, placez le curseur sur LUTXCP2 puis ENTER
- Ensuite Select "Mode Name", F4, placez le curseur sur MODEXCP2 puis ENTER
- Ensuite indiquez le Maximum et Minimun Sessions

* Local LU Name **LUJAS2**
* Remote LU Name **LUTXCP2**
* Mode Name **MODEXCP2**
SESSION Limits
* Maximum Sessions [**4**]
* Minimum First Speaker Sessions [**2**]
* Minimum Bidder Sessions [**2**]

- ENTER pour valider

Mode Initialization Complete.
Limits were negotiated.

- puis F3 pour sortir

- Display Status LUs et Sessions CPI-C/XCP2

- Select -> LU Status Display

Configuration: /usr/xcp2/conf/CONFCGI.Z
LU Status Display

LU	REMOTE LU	MODE	CONFIGURED MAXIMUM	CURRENT MAXIMUM	ACTIVE FSPK	ACTIVE BIDR
LUJAS2	LUTXCP2	SNASVCMG	2	2	1	0
		MODEXCP2	4	4	2	2

- puis F3 pour sortir

- Select -> Display Session Status

Configuration: /usr/xcp2/conf/CONFCGI.Z
Session Status Display

LU	REMOTE	MODE	ID	FSPK	CNVID	TPID	PPID
LUJAS2	LUTXCP2	SNASVCMG	0126404	F			
LUJAS2	LUTXCP2	MODEXCP2	0127780	B			
LUJAS2	LUTXCP2	MODEXCP2	0128428	B			
LUJAS2	LUTXCP2	MODEXCP2	0127052	F			
LUJAS2	LUTXCP2	MODEXCP2	0125732	F			

- puis F3 pour sortir

****

- Définition SIDE INFORMATION

- Revenir sur Menu "Configuration Menu"
- Select -> Side Information Management
-> Update the current SIDE_INFORMATION
-> Add

Symbolic Destination Name *[**CLCFOL**]*

- Partner TP name, indiquez le nom de la transaction du serveur TDS appelant le programme Moniteur de communication
- Ensuite F4 pour lister "Partner LU Name", placez le curseur sur LUTXCP2 puis ENTER
- Ensuite Select "Mode Name", F4, placez le curseur sur MODEXCP2 puis ENTER
- Ensuite indiquez le Path et le nom de la configuration active
- Ensuite Select "Local LU Name", F4, placez le curseur sur LUJAS2 puis ENTER

* Adding Symbolic Destination	**CLCFOL**
* Partner TP name	[**VIC2**]
* Partner LU name (= Remote LU)	[**LUTXCP2**]
* Mode_name	[**MODEXCP2**]

```
  Partner User name                       []
  Partner Password                        []
* Security Type                  0
* Node file path                          [/usr/xcp2/conf/CONFCGI.Z]
* Local LU name                           [LUJAS2]
  BE CAREFUL :
   The environment variable DACSID (in the profile
   file) must contain the PATH of a FILE in which
   is written the PATH of the SIDE INFO FILE.
```

- ENTER pour valider, puis F3 pour sortir

10. Annexe 2 : Utilisation d'un middleware personnalisé

10.1. VisualAge Java

L'utilisation d'un middleware personnalisé consiste à remplacer le middleware de Pacbench Client/Serveur par un middleware spécifique.

Pour mettre en œuvre votre propre middleware, vous devez :

- écraser la classe **`ExchMgrImpl`** par le fichier **`ExchMgrImpl.java`**. Ce fichier contient le code source de cette classe qui implémente le Gestionnaire d'échanges. Il est livré sur le CD-ROM d'installation dans le même répertoire que les fichiers **`vaprun.jar`**, **`vapawt.jar`** et **`vapswing.jar`** correspondant au Runtime Pacbench C/S.
- au sein de cette classe, vous devez redéfinir la méthode **`getNewServer`**.
- et créer une classe qui contient toutes les méthodes implémentant l'interface **`com.ibm.vap.server.Server`**.

10.2. VisualAge Smalltalk

L'utilisation d'un middleware personnalisé consiste à remplacer le middleware de Pacbench Client/Serveur par un middleware spécifique.

Cette opération est possible uniquement dans le contexte de l'utilisation de Pacbench Client/Serveur avec VisualAge.

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 Proxy Vue de Dossier.

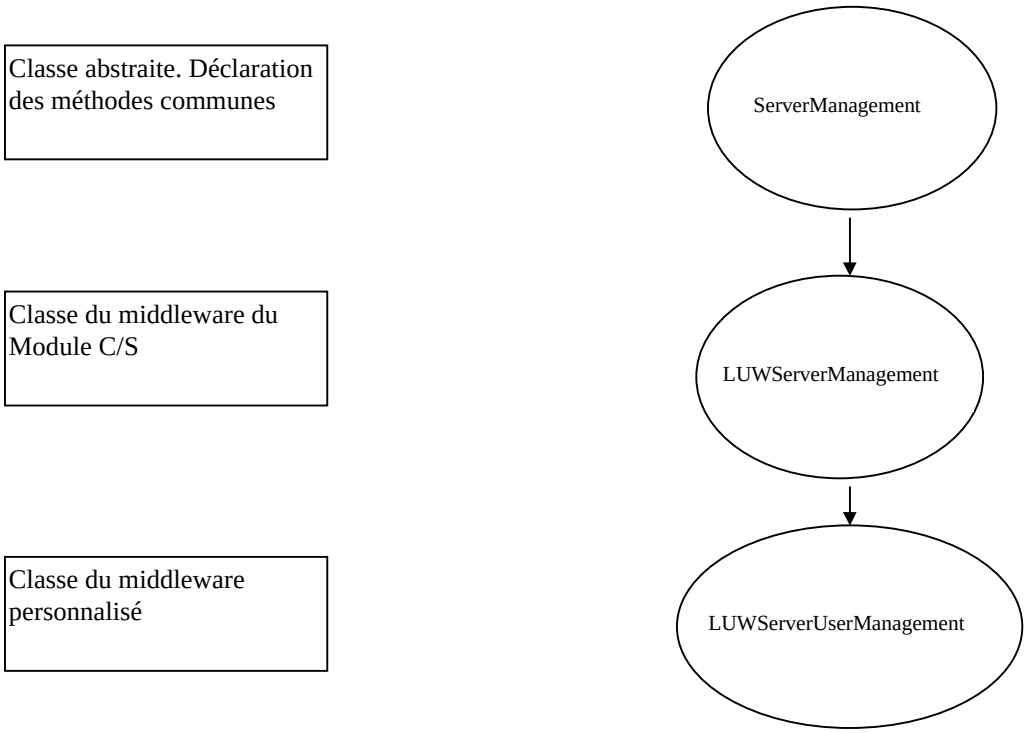
Cette généricité permet de modifier les actions d'appel serveur une seule fois pour que toutes les Proxy Vues de Dossier utilisent le nouveau middleware.

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

10.2.1. Schéma d'héritage des classes d'appel des serveurs

Lorsque le middleware de Pacbench Client/Serveur est utilisé, les actions exécutées proviennent, par mécanisme d'héritage, de la classe **`LUWServerManagement`**.

Lorsque l'on utilise un middleware personnalisé, les actions exécutées correspondent aux actions redéfinies dans la classe **`LUWServerUserManagement`**.



10.2.2. Instanciation des objets du middleware de Pacbench Client/Serveur

La classe **`LUWServerUserManagement`** est à instance unique. L'instanciation et la vérification de l'unicité de son objet sont gérées par la Proxy.

Le choix du type d'objet utilisé pour exécuter une action est effectué par la Proxy en fonction du protocole de communication défini dans le moniteur de communication utilisé par la Proxy.

10.2.3. Actions à surcharger pour l'intégration d'un middleware spécifique

Ces actions correspondent aux points d'entrée qui permettent d'implémenter les services du middleware spécifique.

10.2.3.1. Action callServer: with: with: with

Cette action d'instance appartient à la classe **`LUWServerManagement`**.

Elle est exécutée par une instance de Proxy pour chaque requête émise vers un serveur.

Elle permet d'envoyer un message vers un serveur et d'attendre la réponse pour la transmettre au composant Proxy.

Elle reçoit trois paramètres de type **`string`** correspondant respectivement au code du serveur de communication à appeler, au message à transmettre à ce serveur et au code du buffer local s'il existe.

Elle renvoie un objet **`array`** de deux postes :

- Le premier poste contient un symbol de type **`#OK`** lorsque l'échange s'est correctement effectué ou **`#'Code erreur'`** lorsqu'une erreur de communication a interrompu l'échange.
- Le deuxième poste contient la réponse du serveur lorsque l'échange s'est correctement effectué ou une ordered Collection de messages d'erreur lorsqu'une erreur de communication a interrompu l'échange. Cette liste de messages d'erreur sera automatiquement copiée dans l'attribut **`errorList`** du Gestionnaire d'erreurs ou **`aErrorLabels`** de la Proxy concernée.

10.2.3.2. Création d'un buffer utilisateur local

Ce buffer est utilisé pour transmettre des informations, autres que le code du service appelé et la zone de communication, nécessaires à la couche de communication personnalisée. Ces informations sont transmises à la couche de communication via la Proxy Vue Logique.

Pour générer ce buffer, vous devez coder l'option **`LOCALBUF`** au niveau du Composant Applicatif ou du Dossier Pour des détails, référez-vous au *Guide Utilisateur Pacbench C/S, Vol. II – Services Applicatifs*.

10.2.3.2.1. Part généré correspondant à la description du buffer

Lors de la génération, un buffer local crée un part non-visuel pouvant être appelé sur la Free Form Surface.

Ce buffer est composé de champs que vous alimentez à partir d'autres objets graphiques ou de méthodes Smalltalk spécifiques. Ces champs ne peuvent pas être modifiés par la couche de communication qui ne fait que recevoir les données de ce buffer.

10.2.3.2.2. Attribut LocalBuffer

Cet attribut représente le buffer local. Il est généré dans la classe **ProxyLv**. Il vous permet de connecter le buffer local à la Proxy.

11. INDEX

A

C

E

M

P

V

Cet index ne constitue en aucune manière une liste exhaustive des éléments de l'interface publique, que ce soit pour un Client Java, Smalltalk ou COM.

Pour obtenir cette liste, consultez le *Manuel de Référence Clients Graphiques : Interface Publique des Composants Générés.*