VisualAge Pacbase 2.5

Pacbench Client/Serveur

Manuel de Référence – Clients Graphiques

Interface publique des composants générés

DDOVI000256F

Remarque

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

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

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

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

6ème Edition (novembre 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 pouvez consulter une Table des Matières détaillée.

Table des Matières

Préambule

CONTENU DU MANUEL

Ce manuel décrit exhaustivement l'interface publique des composants générés pour les Clients graphiques de Pacbench C/S en fonction des trois environnements cibles ; Java, Smalltalk et les environnements au standard COM.

L'interface de chaque objet Proxy est générée à partir des caractéristiques – définies dans VisualAge Pacbase – d'une Vue Logique et du Composant Applicatif associé.

L'interface publique d'un objet Proxy est composée de classes, caractérisées par un ensemble d'attributs ou propriétés, d'actions ou méthodes et d'événements. Une application graphique manipule ces éléments de l'interface pour gérer les traitements de chaque Vue Logique selon le Composant Applicatif qui lui est associé.

ORGANISATION DU MANUEL

Ce manuel est constitué en chapitres, suivi d'un *index*.

- Le premier chapitre *Classes*, page 13, présente les classes de l'interface publique, avec les arbres d'héritage.
- Le second chapitre *Attributs*, page 17, donne la liste de tous les types d'attributs sur les différentes plateformes. Pour chaque attribut, le type, le code interne, le nom d'utilisation et le `get`/`set` sont donnés.
- Le troisième chapitre *Actions*, page 60, décrit les actions (ou méthodes pour la plateforme Java) – qu'elles soient locales ou distantes – avec la signature et le nom d'utilisation.
- Le quatrième chapitre *Evénements*, page 105, documente les événements et donne leur code.
- Le cinquième chapitre, *Interface publique de manipulation des Rubriques*, page 111, documente l'API de manipulation des Rubriques.
- Le sixième chapitre, *Interface publique du gestionnaire d'erreurs*, page 117, traite la gestion des erreurs, uniquement dans le cadre des plateformes Smalltalk et COM.
- Le septième chapitre, *Gestion des erreurs pour la cible Java*, page 123, traite la gestion des erreurs Java uniquement.

PREREQUIS ET AUTRES LECTURES

Avant de lire ce volume, vous devez impérativement connaître les grands principes du module Pacbench C/S de VisualAge Pacbase. Ils sont supposés connus dans le présent volume. Consultez le *Guide Utilisateur Pacbench C/S, Vol. I : Concepts – Architectures – Environnement.*

Si vous débutez dans ce type de développement, nous vous conseillons la lecture du *Guide Utilisateur Pacbench C/S, Vol.III : Clients graphiques*. Ce guide vous aidera dans le développement d'un composant Client graphique au travers de nombreux exemples.

CONVENTIONS

La police **`courier`** est utilisée pour toute chaîne de caractères à saisir ou affichés, ou pour des caractères correspondant à du code généré.

La mention "Code interne" désigne le code que vous aurez à saisir en programmation classique.

La mention "Nom d'utilisation" désigne le libellé correspondant affiché dans le Composition Editor, utilisé en programmation visuelle.

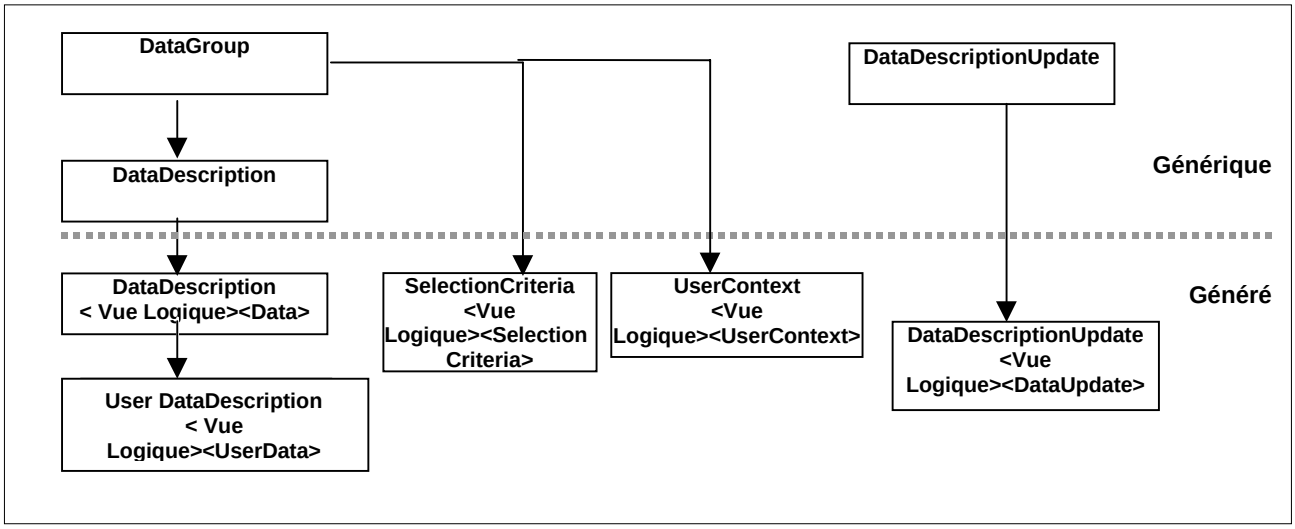
La notation **`[suffixe]`** pour Smalltalk désigne le suffixe choisi pour la classe **`ProxyLv`** au niveau de chaque Composant Applicatif dans VisualAge Pacbase (valeur par défaut = **`ProxyLv`**).

1. Classes

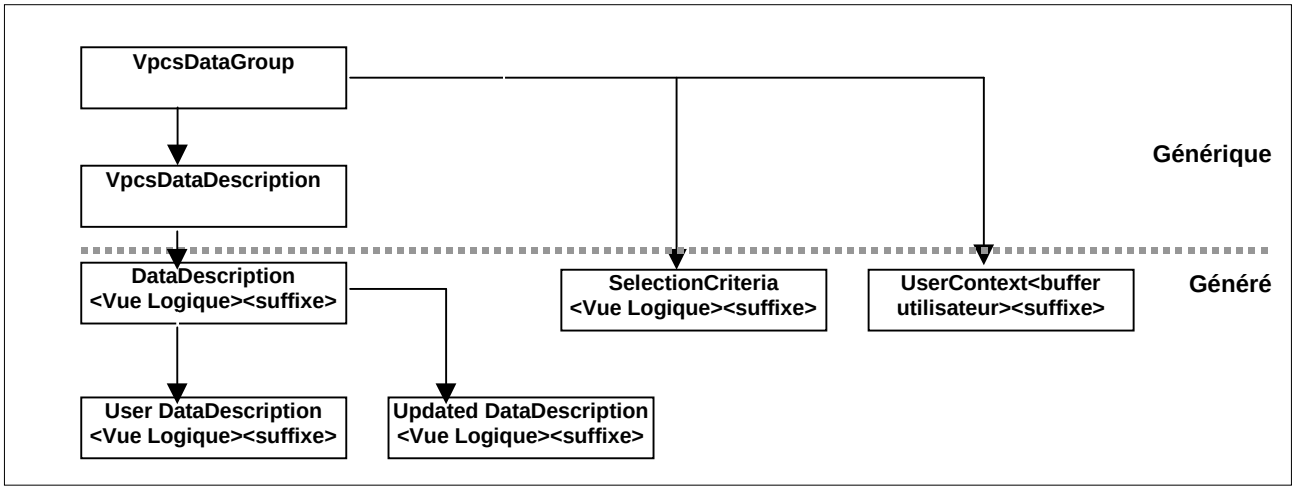
1.1. Classes de données

1.1.1. Schémas d'héritage

1.1.1.1. Java et COM



1.1.1.2. Smalltalk



1.1.2. Classe DataDescription

Cette classe est générée pour chaque Proxy Vue Logique ou nœud d'un Dossier. Elle représente la structure d'une Vue Logique en définissant un attribut par Rubrique de type identifiant ou composition.

Dans le contexte d'un Dossier, la classe **`DataDescription`** associée à un nœud dépendant n'expose pas les Rubriques identifiant des nœuds hiérarchiquement supérieurs.

Une instance de cette classe correspond à une instance de Vue Logique manipulée par l'interface graphique de l'application.

1.1.3. Classe SelectionCriteria

Cette classe est générée pour chaque Proxy Vue Logique ou nœud d'un Dossier. Elle représente la structure de l'identifiant et des paramètres d'extraction d'une Vue Logique en définissant un attribut par Rubrique de type identifiant ou paramètre d'extraction.

Dans le contexte d'un Dossier, la classe **`SelectionCriteria`** associée à un nœud dépendant n'expose pas les Rubriques identifiant des nœuds hiérarchiquement supérieurs.

Il n'existe qu'une seule instance de cette classe par Vue Logique ou par nœud. Cette instance permet de définir l'identifiant et les paramètres d'extraction du début d'une collection ou d'une lecture directe.

1.1.4. Classe DataDescriptionUpdate

Cette classe – appelée également **`UpdatedDataDescription`** dans l'environnement Smalltalk – est générée pour le nœud racine ou dépendant de chaque Dossier. Elle représente la structure d'une Vue Logique modifiée et qualifie son type de modification :

- **Created** : La Vue Logique associée au nœud a été créée localement.
- **Modified** : La Vue Logique associée au nœud a été modifiée localement.
- **Deleted** : La Vue Logique logique associée au nœud a été supprimée localement.
- **Read** : Au moins une des instances dépendantes dans le Dossier a été mise à jour localement.

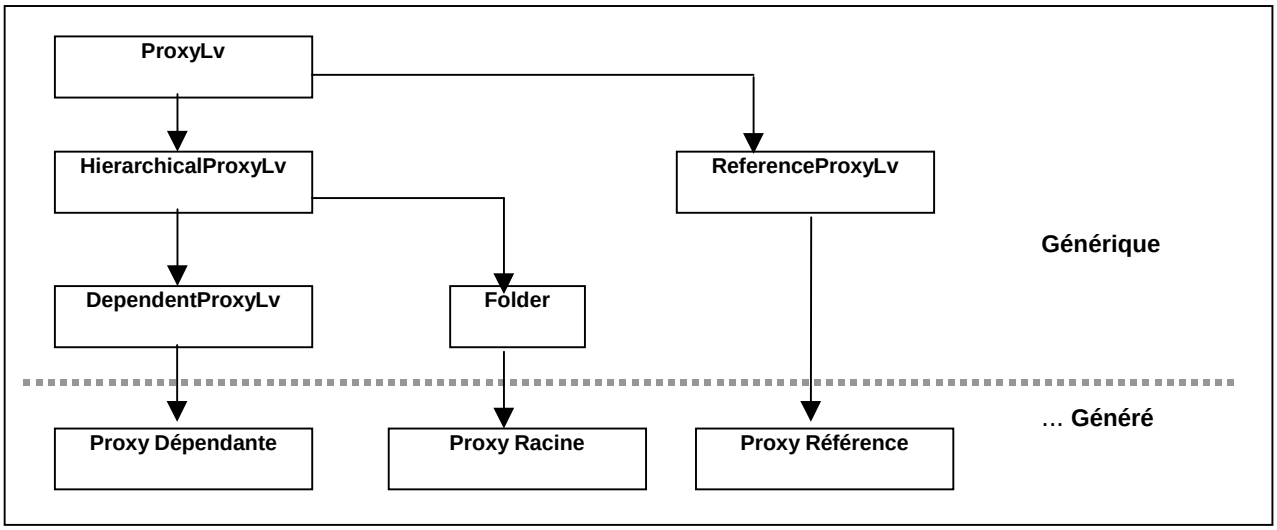
1.1.5. Classe UserContext

Cette classe est générée pour chaque Proxy Vue Logique ou Dossier ou nœud référence d'un Dossier lorsque les serveurs utilisés appellent un buffer utilisateur. Elle représente la structure de ce buffer en définissant un attribut par Rubrique.

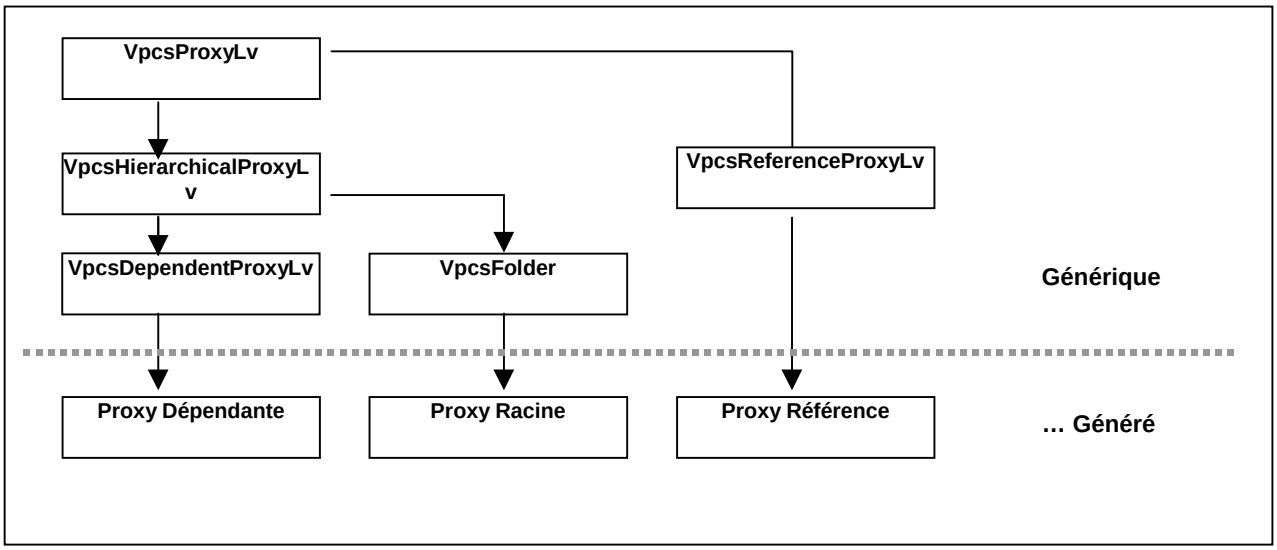
Il n'existe qu'une seule instance de cette classe. Elle est actualisée par l'application graphique ou par les réponses émises par le serveur distant.

1.2. Classe ProxyLv : Schémas d'héritage

1.2.1. Java et COM



1.2.2. Smalltalk



2. Attributs

Un attribut correspond à l'un des trois types d'éléments qui composent l'interface publique d'une classe. Pour les classes publiques associées à une Proxy, il peut correspondre à une constante, à un paramètre ou un résultat d'action. Il est initialisé en fonction du contexte par l'application qui utilise la Proxy ou par la Proxy elle-même.

2.1. Gestion des sélections

2.1.1. Critères de sélection

Description

Cet attribut définit toutes les Rubriques de type identifiant et paramètre d'extraction définis dans la Vue Logique associée à un nœud.

Pour les nœuds dépendants, les Rubriques de type identifiant déjà définies dans le même attribut du nœud père ne sont pas exposées.

L'ordre de description correspond à celui défini dans la Vue Logique.

Chaque Rubrique exposée est initialisée à une valeur vide ou à sa valeur par défaut lorsqu'elle est définie dans le Référentiel.

Cet attribut est systématiquement disponible sur les nœuds de type racine, dépendant ou référence.

Il est accessible en lecture ou en écriture.

Java

- Type `{Nom de classe générée}SelectionCriteria`
- Code interne `selectionCriteria`
- Nom d'utilisation `Selection Criteria`
- get/set `public <Type> selectionCriteria()` / set non disponible

Smalltalk

- Type `SelectionCriteria<CodeVueLogique>[suffixe]`
- Code interne `SelectionCriteria<CodeVueLogique>[suffixe]`
- Nom d'utilisation `selectionCriteria`
- get/set `selectionCriteria` / `selectionCriteria:`

COM

- Type `{Nom de classe générée}SelectionCriteria`
- Code interne `selectionCriteria`
- Nom d'utilisation `Selection Criteria`
- get/set C++ `public <Type> getSelectionCriteria()` / set non disponible

2.1.2. Liste des méthodes d'extraction disponibles

Description

Cet attribut expose la liste des codes de méthodes d'extraction définies dans le Composant Applicatif qui gère la Vue Logique associée au nœud.

Il est disponible sur les nœuds racine, dépendant et référence lorsqu'au moins une méthode d'extraction est définie dans le Composant Applicatif qui gère la Vue Logique associée au nœud.

Il est accessible en lecture uniquement.

Java

- Type `java.lang.String[]`
- Code interne `extractMethodCodes`
- Nom d'utilisation `extraction Method List`
- get/set `public String[] getExtractMethodCodes()` / set non disponible

Smalltalk

- Type `OrderedCollection` contenant `String`
- Code interne `extractMethodList`
- Nom d'utilisation `extractMethodList`
- get/set `extractMethodList` / set non disponible

COM

Disponible avec une API de parcours de collection

- Nb d'éléments `public Int getExtractMethodCodesCount()`
- Elément `public String getExtractMethodCodesElementAt(Int i)`

2.1.3. Méthode d’extraction à exécuter

Description

Cet attribut définit le code de la méthode d’extraction à mettre en œuvre sur une action de sélection de collection.

Il peut prendre une valeur d'initialisation en la définissant dans le panneau de paramétrage de la Proxy.

Il est disponible sur les nœuds racine, dépendant et référence lorsqu’au moins une méthode d’extraction est définie dans le Composant Applicatif qui gère la Vue Logique associée au nœud.

Il est accessible en lecture et en écriture.

Java

- Type `java.lang.String`
- Code interne `extractMethodCode`
- Nom d’utilisation `Extraction Method Code`
- get/set `public String getExtractMethodCode()` / `public void setExtractMethodCode(String s)`

Smalltalk

- Type `String`
- Code interne `extractMethodCode`
- Nom d’utilisation `extractMethodCode`
- get/set `extractMethodCode / extractMethodCode:`

COM

- Type `String`
- Code interne `extractMethodCode`
- Nom d’utilisation `extractMethodCode`
- get/set C++ `public String getExtractMethodCode()` / `public void setExtractMethodCode(String s)`

2.1.4. Mode de gestion de la collection

Description

Cet attribut définit le type de gestion de la collection, au retour d'une action de sélection de collection.

Deux modes dont disponibles :

- Gestion automatique (valeur par défaut)
- Gestion manuelle

La gestion automatique permet, au retour d'une action de sélection de collection, de remplacer la collection courante par les instances sélectionnées.

La gestion manuelle permet, au retour d'une action de sélection de collection, de compléter la collection courante avec les instances sélectionnées. Une instance sélectionnée, déjà présente dans la collection courante, est rafraîchie si l'instance de la collection courante n'a pas été modifiée localement.

Le passage d'un mode à l'autre n'induit aucun changement de la collection courante.

Cet attribut est initialisé à `false` par défaut.

En gestion manuelle, le type de pagination pour les nœuds racines et référence est toujours de type `extend`.

Cet attribut est systématiquement disponible sur les nœuds de type racine, dépendant ou référence.

Il est accessible en lecture ou en écriture.

Java

- Type `Boolean`
- Code interne `manualCollectionReset`
- Nom d'utilisation `manual Collection Reset`
- get/set
```
public Boolean isManualCollectionReset()
public void setManualCollectionReset(Boolean b)
```

Smalltalk

- Type `Boolean`
- Code interne `manualCollectionReset`
- Nom d'utilisation `manualCollectionReset`
- get/set `manualCollectionReset / manualCollectionReset:`

COM

- Type `Boolean`
- Code interne `manualCollectionReset`
- Nom d'utilisation `manualCollectionReset`
- get/set C++
```
public Boolean getManualCollectionReset()
public void setManualCollectionReset(Boolean b)
```

2.2. Gestion des mises à jour

2.2.1. Mise en œuvre des contrôles des données sur le serveur

Description

Cet attribut permet de déclencher les contrôle des données sur le serveur lors de l'exécution d'une méthode de mise à jour serveur.

Il est initialisé à `false` par défaut. Il peut prendre une valeur d'initialisation différente en la définissant dans le panneau de paramétrage de la Proxy.

Il est disponible sur les nœuds racine et dépendant lorsque la Vue Logique associée est utilisée en mise à jour dans son Composant Applicatif et que l'option `CHECKSER` de ce Composant Applicatif est définie.

Il est accessible en lecture et en écriture.

Java

- Type `Boolean`
- Code interne `serverCheckOption`
- Nom d'utilisation `Server Check Option`
- get/set
```
public Boolean getServerCheckOption()
public void setServerCheckOption(Boolean b)
```

Smalltalk

- Type `Boolean`
- Code interne `serverCheckOption`
- Nom d'utilisation `serverCheckOption`
- get/set `serverCheckOption` / `serverCheckOption:`

COM

- Type `Boolean`
- Code interne `serverCheckOption`
- Nom d'utilisation `serverCheckOption`
- get/set C++
```
public Boolean getServerCheckOption()
public void setServerCheckOption(Boolean b)
```

2.2.2. Rafraîchissement des données sur le serveur

Description

Cet attribut permet de récupérer les instances de Vue Logique modifiées en retour d'une mise à jour effectuée par un Composant Applicatif. Cette fonctionnalité concerne essentiellement les Vues Logiques possédant des Rubriques qui sont calculées par le serveur.

Cet attribut est initialisé à `false` par défaut. Il peut prendre une valeur d'initialisation différente en la définissant dans le panneau de paramétrage de la Proxy.

Il est disponible sur les nœuds racine et dépendant lorsque la Vue Logique associée est utilisée en mise à jour dans son Composant Applicatif.

Il est accessible en lecture et en écriture.

Java

- Type `Boolean`
- Code interne `refreshOption`
- Nom d'utilisation `Refresh Option`
- get/set
```
public Boolean getRefreshOption()
public void setRefreshOption(Boolean b)
```

Smalltalk

- Type `Boolean`
- Code interne `refreshOption`
- Nom d'utilisation `refreshOption`
- get/set `refreshOption` / `refreshOption:`

COM

- Type `Boolean`
- Code interne `refreshOption`
- Nom d'utilisation `refreshOption`
- get/set C++
```
public Boolean getRefreshOption()
public void setRefreshOption(Boolean b)
```

2.3. Contrôle du flux des échanges

2.3.1. Limitation du nombre d’instances échangées

Description

Cet attribut définit le nombre maximum d'instances de Vue Logique rendu en un échange par un Composant Applicatif sur une action de récupération de collection.

Il est initialisé à la capacité itérative par défaut de la Vue Logique. Il peut prendre une valeur d'initialisation différente en la définissant dans le panneau de paramétrage de la Proxy.

Il peut prendre une valeur comprise entre 0 et n, n pouvant être supérieur à la capacité itérative de la Vue Logique. Lorsque sa valeur est 0, les actions de sélection simultanée sur plusieurs nœuds ne propagent pas de requête de lecture sur le nœud concerné.

Il est disponible sur les nœuds racine ou référence et sur les nœuds dépendants de cardinalité maximum n.

Il est accessible en lecture et en écriture.

Java

- Type `Integer`
- Code interne `maximumNumberOfRequestedInstances`
- Nom d’utilisation `Maximum Number of Requested Instances`
- get/set
```
public int getMaximumNumberOfRequestedInstances()
public void setMaximumNumberOfRequestedInstances(Int i)
```

Smalltalk

- Type `Integer`
- Code interne `maximumNumberOfRequestedInstances`
- Nom d’utilisation `maximumNumberOfRequestedInstances`
- get/set
```
maximumNumberOfRequestedInstances
maximumNumberOfRequestedInstances:
```

COM

- Type `Integer`
- Code interne `maxNumberOfRequestedInstances`
- Nom d’utilisation `maxNumberOfRequestedInstances`
- get/set C++
```
public int getMaxNumberOfRequestedInstances()
public void setMaxNumberOfRequestedInstances(Int i)
```

2.3.2. Nombre d’instances échangées illimité

Description

Cet attribut permet de récupérer la totalité des instances contenue dans la base de données pour la collection définie par l'action de sélection. Cette fonctionnalité peut générer un nombre important d'échanges entre le composant client et le Composant Applicatif.

Il est initialisé à `false` par défaut. Il peut prendre une valeur d'initialisation différente en la définissant dans le panneau de paramétrage de la Proxy.

Cet attribut est disponible sur les nœuds racine ou référence et sur les nœuds dépendants de cardinalité maximum n.

Il est accessible en lecture et en écriture.

Java

- Type `Boolean`
- Code interne `globalSelection`
- Nom d’utilisation `Global Selection`
- get/set
```
public Boolean getGlobalSelection()
public void setGlobalSelection(Boolean b)
```

Smalltalk

- Type `Boolean`
- Code interne `globalSelectionIndicator`
- Nom d’utilisation `globalSelectionIndicator`
- get/set `globalSelectionIndicator` / `globalSelectionIndicator:`

COM

- Type `Boolean`
- Code interne `globalSelection`
- Nom d’utilisation `globalSelection`
- get/set C++
```
public Boolean getGlobalSelection()
public void setGlobalSelection(Boolean b)
```

2.4. Container d’instances de Vue Logique

2.4.1. Présentation d’une liste d’instances

Description

Cet attribut contient la collection courante du nœud auquel il est associé. Cette collection est constituée d'un ensemble d'instances de Vue Logique. Elle correspond au résultat de la ou des dernières actions de lecture et des actions de mises à jour locales effectuées sur le nœud.

Pour un nœud racine, la collection des instances exposées correspond à la collection de Dossiers récupérée en local.

Pour un nœud dépendant, la collection des instances exposées dépend de l’instance de Vue Logique contenue dans l’attribut de Présentation d’une instance de son nœud père. Les autres instances locales ayant pu être récupérées précédemment sont contenues dans le cache local et seront transférées en fonction des opérations de navigation dans le Dossier.

Pour un nœud référence, la collection des instances exposées correspond à la collection de Vue Logique pouvant être référencée.

Cet attribut est disponible sur les nœuds racine ou référence et sur les nœuds dépendants de cardinalité maximum n.

Il est accessible en lecture uniquement.

Java

- Type `com.ibm.vap.generic.DataDescriptionVector` de `DataDescription` générées
- Code interne `rows`
- Nom d’utilisation `Rows`
- get/set `public DataDescriptionVector rows()`
  set non disponible

Si l’option de génération « `Use IBM EAB classes` » a été choisie, un attribut supplémentaire est généré, afin de faciliter l’utilisation des classes EAB d’IBM :

- Type `COM.ibm.ivj.javabeans.IVector`
- Code interne `iRows`
- Nom d’utilisation `IRows`
- get/set `public COM.ibm.ivj.javabeans.IVector iRows()`
  set non disponible

Smalltalk

- Type `VpcsOrderedCollection` contenant `DataDescription<codeVueLogique>[suffixe]`
- Code interne `rows`
- Nom d’utilisation `rows`
- get/set `rows` / set non disponible

COM

Disponible avec une API de parcours de collection

- Nb d’éléments `public Int getRowsCount()`
- Elément `public {Nom de classe générée}Data getRowsElementAt(Int i)`

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

Description

Cet attribut permet de spécifier si la collection contenue dans l'attribut de Présentation d'une liste d'instances (page 25) doit être triée selon le critère de tri local (`true`) ou serveur (`false`).

Par défaut, les instances contenues dans l'attribut de Présentation d'une liste d'instances sont triées selon le critère de tri local.

Cet attribut est disponible sur les nœuds racine ou référence et sur les nœuds dépendants de cardinalité maximum n.

Cet attribut est accessible en lecture et en écriture.

Java

- Type `Boolean`
- Code interne `localSort`
- Nom d'utilisation `localSort`
- get/set `public boolean getLocalSort()` / `public setLocalSort(boolean)`

Smalltalk

- Type `Boolean`
- Code interne `localSort`
- Nom d'utilisation `localSort`
- get/set `localSort` / `localSort:`

COM

- Type `Boolean`
- Code interne `localSort`
- Nom d'utilisation `localSort`
- get/set
```
public Boolean isLocalSort()
public void setLocalSort(Boolean a)
```

2.4.3. Critère local de tri d’une liste d’instances

Description

Cet attribut permet de définir le critère de tri local appliqué à la collection contenue dans l'attribut de Présentation d’une liste d’instances, page 25.

Par défaut, les instances contenues dans l'attribut de Présentation d’une liste d’instances sont triées dans l'ordre de l'identifiant de la Vue Logique.

Cet attribut est disponible sur les nœuds racine ou référence et sur les nœuds dépendants de cardinalité maximum n.

Cet attribut est accessible en lecture et en écriture.

Java

En Java, le critère de tri est représenté par une instance d’une classe implémentant l’interface générique `com.ibm.vap.generic.Comparator`. Cette interface est constituée de la signature de méthode suivante :

```
public int compare(Object a, Object b) ;
```

L’implémentation de cette méthode doit fournir un critère de tri permettant de placer `a` avant `b`, si `compare` a rendu un nombre négatif, et `b` avant `a` sinon.

- Type `com.ibm.vap.generic.Comparator`
- Code interne `dataComparator`
- Nom d’utilisation -
- get/set
```
public com.ibm.vap.generic.Comparator getDataComparator()
public void setDataComparator(com.ibm.vap.generic.Comparator c)
```

Smalltalk

La syntaxe utilisée en Smalltalk pour définir un critère de tri est la suivante : **`[ :a :b |`** ***condition*** **`]`** où a et b représentent deux instances quelconques de la liste des instances. La liste des instances est alors ordonnée afin que tous les couples d’instances constitués à partir de la liste satisfassent à la condition .

Par exemple, le tri par défaut est réalisé par le block suivant : **`[ :a :b | a ident <= b ident ]`**où ident représente l’identifiant de la Vue Logique. La liste est alors dans l’ordre croissant des identifiants de la Vue Logique.

- Type `BlockContextTemplate`
- Code interne `rowsSortBlock`
- Nom d’utilisation `rowsSortBlock`
- get/set `rowsSortBlock` / `rowsSortBlock:`

COM

Non disponible.

2.4.4. Présentation d’une instance

Description

Cet attribut permet d'exposer une instance de Vue Logique. Il définit toutes les Rubriques de la Vue Logique qui ne sont pas définies comme paramètre d'extraction.
Pour les nœuds dépendants, les Rubriques qui définissent l'identifiant de la hiérarchie supérieure ne sont pas exposées. L'initialisation de ces Rubriques est gérée automatiquement par la Proxy Vue de Dossier en fonction des instances courantes contenues dans les nœuds hiérarchiques supérieurs.

Lorsque cet attribut est vide, chaque Rubrique exposée est initialisée à une valeur vide ou à sa valeur par défaut lorsqu'elle est définie dans le Référentiel.

Après chaque action de lecture directe ou de lecture de collection ne retournant qu'une instance, cet attribut est initialisé par l'instance de Vue Logique récupérée du serveur.

Cet attribut est systématiquement disponible sur les nœuds racine, dépendant et référence.

Il est accessible en lecture ou en écriture.

Java

- Type `{Nom de classe générée}Data`
- Code interne `detail`
- Nom d’utilisation `Detail`
- get/set `public <Type généré> detail()` / set non disponible

Smalltalk

- Type `DataDescription<CodeVueLogique>[suffixe]`
- Code interne `detail`
- Nom d’utilisation `detail`
- get/set `detail` / `getDetailFromDataDescription: aDataDescription`

COM

- Type `{Nom de classe générée}Data`
- Code interne `detail`
- Nom d’utilisation `detail`
- get/set C++ `public <Type> getDetail()` / set non disponible

2.4.5. Présentation des Dossiers modifiés

Description

Cet attribut permet d'exposer la liste des Dossiers qui ont été modifiés localement. Il permet par exemple d'exécuter une action d'annulation des modifications locales sur une instance de Dossier supprimée qui n'apparaît plus dans l'attribut de Présentation d'une liste d'instances.

Il expose pour chaque Dossier modifié :

- L'instance de Vue Logique du nœud racine du Dossier
- Le nombre de services de modification associés à l'instance de Dossier modifiée
- Le statut de modification qui peut prendre les valeurs suivantes :

#Created	instance créée localement
#Modified	instance modifiée localement
#Deleted	instance effacée localement
#Read	instance lue, mais dont certaines instances dépendantes ont été mises à jour localement.

Cet attribut est disponible sur le nœud racine d'une Proxy Vue de Dossier lorsque les options définies dans les Composants Applicatifs qui gèrent le Dossier lui permettent d'être modifiable.

Il est accessible en lecture uniquement.

Java

- Type `com.ibm.vap.generic.DataDescriptionUpdateVector` de `DataDescriptionUpdate`
- Code interne `updatedFolders`
- Nom d'utilisation `Updated Folders`
- get/set `public DataDescriptionUpdateVector updatedFolders()`
  set non disponible

Si l'option de génération « `Use IBM EAB classes` » a été choisie, un attribut supplémentaire est généré, afin de faciliter l'utilisation des classes EAB d'IBM :

- Type `COM.ibm.ivj.javabeans.IVector`
- Code interne `iUpdatedFolders`
- Nom d'utilisation `IUpdated Folders`
- get/set `public COM.ibm.ivj.javabeans.IVector iUpdatedFolders()`
  set non disponible

Smalltalk

- Type `VpcsOrderedCollection` contenant `UpdatedDataDescription<CodeVueLogique>[suffixe]`
- Code interne `updatedFolders`
- Nom d'utilisation `updatedFolders`
- get/set `updatedFolders` / set non disponible

COM

Disponible avec une API de parcours de collection

- Nb d'éléments `public Int getUpdatedFoldersCount()`
- Elément `public {Nom de classe générée}DataUpdate getUpdatedFoldersElementAt(Int i)`

2.4.6. Présentation des instances modifiées

Description

Cet attribut permet d'exposer la liste des instances du noeud qui ont été modifiées localement. Il permet par exemple d'exécuter une action d'annulation des modifications locales sur une instance supprimée qui n'apparaît plus dans l'attribut de Présentation d'une liste d'instances.

Il expose pour chaque noeud modifié :

- L'instance de Vue Logique du nœud
- Le nombre de services de modification associés à l'instance modifiée
- Le statut de modification qui peut prendre les valeurs suivantes :
  #Created instance créée localement
  #Modified instance modifiée localement
  #Deleted instance effacée localement
  #Read instance lue, mais dont certaines instances dépendantes ont été mises à jour localement.

Cet attribut est disponible sur un nœud racine ou dépendant d'une Proxy Vue de Dossier lorsque le Composant Applicatif associé au noeud comporte un service de mise à jour.

Il est accessible en lecture uniquement.

Java

- Type `com.ibm.vap.generic.DataDescriptionUpdateVector` de `DataDescriptionUpdate`
- Code interne `updatedInstances`
- Nom d'utilisation `Updated Instances`
- get/set `public DataDescriptionUpdateVector updatedInstances()`
  set non disponible

Si l'option de génération « `Use IBM EAB classes` » a été choisie, un attribut supplémentaire est généré, afin de faciliter l'utilisation des classes EAB d'IBM :

- Type `COM.ibm.ivj.javabeans.IVector`
- Code interne `iUpdatedInstances`
- Nom d'utilisation `IUpdated Instances`
- get/set `public COM.ibm.ivj.javabeans.IVector iUpdatedInstances()`
  set non disponible

Smalltalk

- Type `VpcsOrderedCollection` contenant `UpdatedDataDescription<CodeVueLogique>[suffixe]`
- Code interne `updatedInstances`
- Nom d'utilisation `updatedInstances`
- get/set `updatedInstances` / set non disponible

COM

Disponible avec une API de parcours de collection

- Nb d'éléments `public Int getUpdatedInstancesCount()`
- Elément `public {Nom de classe générée}DataUpdate getUpdatedInstancesElementAt(Int i)`

2.4.7. Présentation d’instances destinées à un Service Utilisateur

Description

Cet attribut contient une liste d'instances de Vue Logique créée par les actions locales de préparation d’un Service Utilisateur. Ces instances sont indépendantes des instances contenues dans l'attribut de Présentation d’une liste d’instances, page 25.

Les règles de présentation des instances en fonction de la hiérarchie des nœuds ne sont pas appliquées sur cet attribut.

Ces instances seront envoyées au serveur à l'exécution de la prochaine action de soumission d'un Service Utilisateur serveur.

Cet attribut est disponible sur les nœuds racine ou dépendant lorsqu’au moins un Service Utilisateur est défini dans le Composant Applicatif qui gère la Vue Logique et que celle-ci dispose d’une capacité itérative supérieure à 1.

Cet attribut est accessible en lecture uniquement.

Java

- Type `com.ibm.vap.generic.DataDescriptionVector` de `UserDataDescription` générées
- Code interne `userInputRows`
- Nom d’utilisation `User Input Rows`
- get/set `public DataDescriptionVector userInputRows()`
  set non disponible

Si l’option de génération « `Use IBM EAB classes` » a été choisie, un attribut supplémentaire est généré, afin de faciliter l’utilisation des classes EAB d’IBM :

- Type `COM.ibm.ivj.javabeans.IVector`
- Code interne `iUserInputRows`
- Nom d’utilisation `IUser Input Rows`
- get/set `public COM.ibm.ivj.javabeans.IVector iUserInputRows()`
  set non disponible

Smalltalk

- Type `VpcsOrderedCollection` contenant `UserDataDescription<CodeVueLogique>[suffixe]`
- Code interne `userServiceInputRows`
- Nom d’utilisation `userServiceInputRows`
- get/set `userServiceInputRows` / set non disponible

COM

Disponible avec une API de parcours de collection

- Nb d’éléments `public Int getUserInputRowsCount()`
- Elément `public {Nom de classe générée}UserData getUserInputRowsElementAt(Int i)`

2.4.8. Présentation d'instances rendues par un Service Utilisateur

Description

Cet attribut contient une liste d'instances de Vue Logique renvoyée par le Service Utilisateur serveur après son exécution. Ces instances sont indépendantes des instances contenues dans l'attribut de Présentation d'une liste d'instances, page 25.

Les règles de présentation des instances en fonction de la hiérarchie des nœuds ne sont pas appliquées sur cet attribut.

Cet attribut est disponible sur les nœuds racine ou dépendant lorsqu'au moins un Service Utilisateur est défini dans le Composant Applicatif qui gère la Vue Logique et que celle-ci dispose d'une capacité itérative supérieure à 1.

Cet attribut est accessible en lecture et en écriture.

Java

- Type : `com.ibm.vap.generic.DataDescriptionVector` de `UserDataDescription` générées
- Code interne : `userOutputRows`
- Nom d'utilisation : `User Output Rows`
- get/set : `public DataDescriptionVector userOutputRows()` set non disponible

Si l'option de génération « `Use IBM EAB classes` » a été choisie, un attribut supplémentaire est généré, afin de faciliter l'utilisation des classes EAB d'IBM :

- Type : `COM.ibm.ivj.javabeans.IVector`
- Code interne : `iUserOutputRows`
- Nom d'utilisation : `IUser Output Rows`
- get/set : `public COM.ibm.ivj.javabeans.IVector iUserOutputRows()` set non disponible

Smalltalk

- Type : `VpcsOrderedCollection` contenant `UserDataDescription<CodeVueLogique>[suffixe]`
- Code interne : `userServiceOutputRows`
- Nom d'utilisation : `userServiceOutputRows`
- get/set : `userServiceOutputRows` / `userServiceOutputRows:`

COM

Disponible avec une API de parcours de collection

- Nb d'éléments : `public Int getUserOutputRowsCount()`
- Elément : `public {Nom de classe générée}UserData getUserOutputRowsElementAt(Int i)`

2.4.9. Présentation d’une instance liée à un Service Utilisateur

Description

Cet attribut permet d'exposer une instance de Vue Logique à transmettre ou renvoyée par un Service Utilisateur serveur. Il définit toutes les Rubriques de la Vue Logique qui ne sont pas définies comme paramètre d'extraction.

Les règles de présentation des instances en fonction de la hiérarchie des nœuds ne sont pas appliquées sur cet attribut. En conséquence, dans un nœud dépendant, les Rubriques qui définissent l'identifiant de la hiérarchie supérieure sont exposées.

Lorsque cet attribut est vide, chaque Rubrique exposée est initialisée à une valeur vide ou à sa valeur par défaut lorsqu'elle est définie dans le Référentiel.

Lorsqu'un Service Utilisateur de type serveur ne renvoie qu'une instance de Vue Logique, cet attribut l'expose automatiquement.

Cet attribut est disponible sur les nœuds racine ou dépendant lorsqu’au moins un Service Utilisateur est défini dans le Composant Applicatif qui gère la Vue Logique associée au nœud.

Il est accessible en lecture uniquement.

Java

- Type `{Nom de Classe générée}UserData`
- Code interne `userDetail`
- Nom d’utilisation `User Detail`
- get/set `public {Nom de Classe générée}UserData userDetail()`
  set non disponible

Smalltalk

- Type `UserDataDescription<CodeVueLogique>[suffixe]`
- Code interne `userDetail`
- Nom d’utilisation `userDetail`
- get/set `userDetail` / set non disponible

COM

- Type `{Nom de Classe générée}UserData`
- Code interne `userDetail`
- Nom d’utilisation `userDetail`
- get/set C++ `public <Type> getUserDetail()`/ set non disponible

2.5. Gestion des Services Utilisateur

2.5.1. Liste des Services Utilisateur disponibles

Description

Cet attribut expose la liste des codes des Services Utilisateur définis dans le serveur qui gère la Vue Logique associée au nœud.

Cet attribut est disponible sur les nœuds racine ou dépendant lorsqu'au moins un Service Utilisateur est défini dans le Composant Applicatif qui gère la Vue Logique associée au nœud.

Il est accessible en lecture uniquement.

Java

- Type `java.lang.String[]`
- Code interne `userServiceCodes`
- Nom d'utilisation `User Service Codes`
- get/set `public String[] getUserServiceCodes()`
  set non disponible

Smalltalk

- Type `OrderedCollection` contenant `String`
- Code interne `userServiceList`
- Nom d'utilisation `userServiceList`
- get/set `userServiceList` / non disponible

COM

Disponible avec une API de parcours de collection

- Nb d'éléments `public Int getUserServiceCodesCount()`
- Elément `public String getUserServiceCodesElementAt(Int i)`

2.5.2. Services Utilisateur à exécuter

Description

Cet attribut permet de définir le code du Service Utilisateur qui sera traité sur le serveur qui gère le nœud lorsque l'action d'exécution des Services Utilisateur sera déclenchée sur le nœud racine d'un Dossier.

Il est disponible sur les nœuds racine ou dépendant lorsqu'au moins un Service Utilisateur est défini dans le Composant Applicatif qui gère la Vue Logique associée au nœud.

Il est accessible en lecture et en écriture.

Java

- Type `java.lang.String`
- Code interne `userServiceCode`
- Nom d'utilisation `User Service Code`
- get/set
```
public String getUserServiceCode()
public void setUserServiceCode(String s)
```

Smalltalk

- Type `String`
- Code interne `userServiceCode`
- Nom d'utilisation `userServiceCode`
- get/set `userServiceCode` / `userServiceCode:`

COM

- Type `String`
- Code interne `userServiceCode`
- Nom d'utilisation `userServiceCode`
- get/set C++
```
public String getUserServiceCode()
public void setUserServiceCode(String s)
```

2.6. Gestion des verrouillages logiques

2.6.1. Identifiant de verrouillage d’un Dossier

Description

Cet attribut expose une chaîne de caractères calculée par le serveur et rendue à la dernière demande de verrouillage logique réussie d’une instance de Dossier. Il est associé à l'instance courante de Dossier contenue dans le nœud racine.

Cet attribut est disponible sur le nœud racine d’une Proxy Vue de Dossier lorsque l’option ‘verrouillage logique’ du Dossier est positionnée.

Lorsque l'option de verrouillage logique d'un Dossier est positionnée et que cet attribut est vide, toute mise à jour locale d'un nœud quelconque du Dossier est refusée.

Cet attribut est initialisé automatiquement à une valeur vide pour toutes les instances de Dossier concernées par une mise à jour serveur réussie ou lorsqu'une action de déverrouillage logique explicite a été exécutée.

Cet attribut est accessible en lecture uniquement.

Aucun événement n’est émis lorsque la valeur de cet attribut change. Il est donc déconseillé d’utiliser des connexions attribut à attribut ou événement à méthode avec cet attribut.

Java

- Type `String`
- Code interne `lockTimestamp`
- Nom d’utilisation `Lock Timestamp`
- get/set `public String getLockTimestamp()` / set non disponible

Smalltalk

- Type `String`
- Code interne `lockTimeStamp`
- Nom d’utilisation `lockTimeStamp`
- get/set `lockTimeStamp` / set non disponible

COM

- Type `String`
- Code interne `getLockTimestamp`
- Nom d’utilisation `lockTimestamp`
- get/set C++ `public String getLockTimestamp()` / set non disponible

2.7. Gestion des messages de retour de sélection

2.7.1. Libellé du message

Description

Cet attribut expose le libellé d'un message d'information renvoyé par un serveur à la suite de l'exécution d'une action de sélection lorsque la fin de la collection demandée est atteinte ou qu'une instance demandée n'est pas trouvée ou est incomplète.

Cet attribut est systématiquement disponible sur tous les types de nœud.

Il est accessible en lecture uniquement.

Java

- Type `java.lang.String`
- Code interne `accessInfoLabel`
- Nom d’utilisation `Access Info Label`
- get/set `public String getAccessInfoLabel()`
  set non disponible

Smalltalk

- Type `String`
- Code interne `accessInfoLabel`
- Nom d’utilisation `accessInfoLabel`
- get/set `accessInfoLabel` / set non disponible

COM

- Type `String`
- Code interne `accessInfoLabel`
- Nom d’utilisation `accessInfoLabel`
- get/set C++ `public String getAccessInfoLabel()`/ set non disponible

2.7.2. Clé du message

Description

Cet attribut expose la clé d'un message d'information renvoyé par un serveur à la suite de l'exécution d'une action de sélection lorsque la fin de la collection demandée est atteinte ou qu'une instance demandée n'est pas trouvée ou est incomplète.
Cet attribut est systématiquement disponible sur tous les types de nœud.

Il est accessible en lecture uniquement.

Java

- Type `java.lang.String`
- Code interne `accessInfoKey`
- Nom d’utilisation `Access Info Key`
- get/set `public String getAccessInfoKey()`
  set non disponible

Smalltalk

- Type `String`
- Code interne `accessInfoKey`
- Nom d’utilisation `accessInfoKey`
- get/set `accessInfoKey` / set non disponible

COM

- Type `String`
- Code interne `accessInfoKey`
- Nom d’utilisation `accessInfoKey`
- get/set C++ `public String getAccessInfoKey()`
  set non disponible

2.8. Gestion des messages d'erreur

2.8.1. Objet erreur

Description

Cet attribut contient une instance d'erreurs renvoyée à la suite d'une action appliquée sur un ou plusieurs nœuds du Dossier.

Cet attribut est systématiquement disponible sur le nœud racine.

Il est accessible en lecture ou en écriture.

Java

Non disponible.
Les erreurs sont générées sous la forme d'exceptions levées par les actions des Proxies.

Pour plus de précision sur la gestion des erreurs en langage Java, reportez-vous au chapitre 7, *Gestion des erreurs pour la cible Java*, page 123

Smalltalk

- Type `VpcsErrorManager`
- Code interne `errorManager`
- Nom d'utilisation `errorManager`
- get/set `errorManager` / `errorManager:`

COM

Non disponible.

2.9. Gestion des événements

2.9.1. Liste des événements de la dernière action serveur

Description

Cet attribut contient un tableau de constantes entières correspondant aux différents événements renvoyés par la dernière action serveur.

Cet attribut est systématiquement disponible sur le nœud racine.

Il est accessible en lecture.

Java

Non disponible

Smalltalk

Non disponible.

COM

Disponible avec une API de gestion d'une pile. La méthode de récupération d'un événement récupère le premier événement de la pile et l'élimine de celle-ci.

- Nb d'éléments `public Int getServerEventsCount()`
- Elément `public String popServerEvent()`

2.10. Gestion des informations contextuelles

2.10.1. Informations contextuelles

Description

Cet attribut contient les Rubriques d'une structure d'information contextuelle envoyée et reçue à chaque exécution d'une action serveur associée à un nœud racine ou dépendant.
Cet attribut est disponible lorsqu'un buffer utilisateur a été défini au niveau du Dossier. Il est accessible en lecture uniquement.

Java

- Type `{Nom de Classe générée}Buffer`
- Code interne `folderUserContext`
- Nom d'utilisation `Folder User Context`
- get/set `public {Nom de Classe générée}Buffer folderUserContext()`
  set non disponible

Smalltalk

- Type `UserContext<code buffer utilisateur>[suffixe]`
- Code interne `folderUserContext`
- Nom d'utilisation `folderUserContext`
- get/set `folderUserContext` / set non disponible

COM

- Type `{Nom de Classe générée}Buffer`
- Code interne `folderUserContext`
- Nom d'utilisation `folderUserContext`
- get/set C++ `public <Type> getFolderUserContext()` / set non disponible

2.10.2. Informations contextuelles liées aux nœuds référence

Description

Cet attribut contient les Rubriques d'une structure d'information contextuelle envoyée et reçue à chaque exécution d'une action serveur associée à un nœud référence.
Cet attribut est disponible lorsqu'un buffer utilisateur a été défini au niveau du Composant Applicatif gérant le nœud référence. Il est accessible en lecture uniquement.

Java

- Type `{Nom de Classe générée}Buffer`
- Code interne `referenceUserContext`
- Nom d'utilisation `Reference User Context`
- get/set `public {Nom de Classe générée}Buffer referenceUserContext()`/ set non disponible

Smalltalk

- Type `UserContext<code buffer utilisateur>[suffixe]`
- Code interne `referenceUserContext`
- Nom d'utilisation `referenceUserContext`
- get/set `referenceUserContext` / set non disponible

COM

- Type `{Nom de Classe générée}Buffer`
- Code interne `referenceUserContext`
- Nom d'utilisation `referenceUserContext`
- get/set C++ `public <Type> getReferenceUserContext()`/ set non disponible

2.11. Compteurs disponibles

2.11.1. Nombre total d'instances locales

Description

Cet attribut contient le nombre d'instances locales contenues dans le cache de la Proxy Vue de Dossier, tous nœuds confondus.

Cet attribut est systématiquement disponible sur le nœud racine. Il est accessible en lecture uniquement.

Java

- Type `int`
- Code interne `folderInstancesCount`
- Nom d'utilisation `Folder Instances Count`
- get/set `public int getFolderInstancesCount()`
  set non disponible

Smalltalk

- Type `Integer`
- Code interne `folderInstancesCount`
- Nom d'utilisation `folderInstancesCount`
- get/set `folderUpdatedInstancesCount` / set non disponible

COM

Cette information n'est pas référencée en tant qu'attribut dans l'API de la Proxy. Elle est disponible en exécutant la méthode suivante :

- get/set C++ `public int getFolderInstancesCount()` / set non disponible

Nombre total de services de mise à jour

Description

Cet attribut contient le nombre de mises à jour qui seront effectuées sur les différents serveurs de Vue Logique à la prochaine émission de l'action d'exécution des mises à jour serveur. Ce nombre concerne toutes les instances de Dossier modifiées, tous nœuds confondus.
Cet attribut est disponible sur le nœud racine lorsqu'au moins une Vue Logique associée à l'un des nœuds du Dossier est utilisée en mise à jour dans le Composant Applicatif la gérant.

Cet attribut est accessible en lecture uniquement.

Java

- Type `int`
- Code interne `folderUpdatedInstancesCount`
- Nom d'utilisation `Folder Updated Instances Count`
- get/set `public int getFolderUpdatedInstancesCount()`
  set non disponible

Smalltalk

- Type `Integer`
- Code interne `folderUpdatedInstancesCount`
- Nom d'utilisation `folderUpdatedInstancesCount`
- get/set `folderUpdatedInstancesCount` / set non disponible

COM

Cette information n'est pas référencée en tant qu'attribut dans l'API de la Proxy. Elle est disponible en exécutant la méthode suivante :

- get/set C++ `public int getFolderUpdatedInstancesCount()`
  set non disponible

2.11.2. Nombre de services de mise à jour associés à un nœud

Description

Cet attribut contient le nombre de mises à jour qui seront effectuées sur le serveur associé au nœud à la prochaine émission de l'action d'exécution des mises à jour serveur.
Cet attribut est disponible sur chaque nœud racine et dépendant dont la Vue Logique associée est en mise à jour dans le Composant Applicatif la gérant.

Il est accessible en lecture uniquement.

Java

- Type `int`
- Code interne `nodeUpdatedInstancesCount`
- Nom d'utilisation `Node Updated Instances Count`
- get/set `public int getNodeUpdatedInstancesCount()`
  set non disponible

Smalltalk

- Type `Integer`
- Code interne `nodeUpdatedInstancesCount`
- Nom d'utilisation `nodeUpdatedInstancesCount`
- get/set `nodeUpdatedInstancesCount` / set non disponible

COM

Cette information n'est pas référencée en tant qu'attribut dans l'API de la Proxy. Elle est disponible en exécutant la méthode suivante :

- get/set C++ `public int getNodeUpdatedInstancesCount()`
  set non disponible

2.11.3. Nombre d'instances de Vue Logique réservées à un Service Utilisateur

Description

Cet attribut contient le nombre d'instances de Vue Logique de l'ensemble des nœuds du Dossier qui seront envoyées sur le serveur par une action d'exécution des Services Utilisateur.

Cet attribut est disponible sur un nœud racine si au moins un Service Utilisateur est défini sur le Composant Applicatif gérant la Vue Logique du nœud racine ou d'un de ses dépendants.

Il est accessible en lecture uniquement.

Java

Non disponible.

Smalltalk

- Type `Integer`
- Code interne `folderInputUserServiceInstancesCount`
- Nom d'utilisation `folderInputUserServiceInstancesCount`
- get/set `folderInputUserServiceInstancesCount` / set non disponible

COM

Non disponible.

2.11.4. Nombre d’instances de Vue Logique traitées par un Service Utilisateur

Description

Cet attribut contient le nombre d’instances de Vue Logique, tous nœuds confondus, renvoyées par les serveurs suite à une action d’exécution de Services Utilisateur.

Cet attribut est disponible sur un nœud racine si au moins un Service Utilisateur est défini sur le Composant Applicatif gérant la Vue Logique du nœud racine ou d’un de ses dépendants.

Il est accessible en lecture uniquement.

Java

Non disponible.

Smalltalk

- Type `Integer`
- Code interne `folderOutputUserServiceInstancesCount`
- Nom d’utilisation `folderOutputUserServiceInstancesCount`
- get/set `folderOutputUserServiceInstancesCount` / set non disponible

COM

Non disponible.

2.12. Gestion des communications

2.12.1. Liste des plateformes disponibles

Description

Cet attribut contient la liste des codes logiques (localisations) de toutes les plateformes d'exécution disponibles pour un Dossier.

Cet attribut est systématiquement disponible sur le nœud racine.

Il est accessible en lecture ou en écriture.

Java

Lors de l'utilisation de la version Java ou COM avec une gateway, la première exécution de `getLocations` pour Java et `getLocationsCount` pour COM (soit directement, soit lors du premier accès serveur) provoque l'appel d'un service spécifique de la gateway, dont le rôle est de renvoyer la liste des noms logiques de localisation associés au Dossier.

- Type `java.lang.String[]`
- Code interne `locations`
- Nom d'utilisation `Locations`
- get/set `public String[] getLocations()`
  set non disponible

Smalltalk

- Type `VpcsOrderedCollection` contenant `VpcsLocation`
- Code interne `locationList`
- Nom d'utilisation `locationList`
- get/set `locationList` / `locationList:`

COM

Disponible avec une API de parcours de collection

- Nb d'éléments `public Int getLocationsCount()`
- Elément `public String getLocationsElementAt(Int i)`

2.12.2. Plateforme sélectionnée pour l’exécution d’une requête

Description

Cet attribut contient le code logique (localisation) du prochain service à exécuter sur le serveur.

La population présente dans le cache local n'est pas réinitialisée lors d'un changement de location. Il est cependant possible, le cas échéant, d'éliminer du cache local toutes les instances du nœud et de ses dépendants par l'action resetCollection ; voir le paragraphe Initialisation de la collection, page 76.

Cet attribut est systématiquement disponible sur le nœud racine.

Il est accessible en lecture ou en écriture.

Java

- Type `String`
- Code interne `location`
- Nom d’utilisation `Location`
- get/set
```
public String getLocation()
public void setLocation(String l)
```

Smalltalk

- Type `VpcsLocation`
- Code interne `location`
- Nom d’utilisation `location`
- get/set `location` / `location:`

COM

- Type `String`
- Code interne `location`
- Nom d’utilisation `location`
- get/set C++
```
public String getLocation() /
public void setLocation(String l)
```

2.12.3. Code utilisateur de connexion à la plateforme

Description

Cet attribut contient le code utilisateur de connexion à la plateforme d'exécution sélectionnée.

Cet attribut est systématiquement disponible sur le nœud racine.

Il est accessible en lecture / écriture.

Java

- Type `String`
- Code interne `userId`
- Nom d'utilisation `User Id`
- get/set get non disponible
  `public void setUserId(String u)`

Smalltalk

- Type `VpcsUserIdentification`
- Code interne `userId`
- Nom d'utilisation `userId`
- get/set `userId` / `userId:`

COM

- Type `String`
- Code interne `userId`
- Nom d'utilisation `userId`
- get/set C++ get non disponible / `public void setUserId(String u)`

2.12.4. Mot de passe de connexion à la plateforme

Description

Cet attribut contient le mot de passe du code utilisateur de connexion à la plateforme d'exécution sélectionnée.

Cet attribut est systématiquement disponible sur le nœud racine.

Il est accessible en lecture / écriture.

Java

- Type `String`
- Code interne `password`
- Nom d'utilisation `Password`
- get/set get non disponible
  `public void setPassword(String p)`

Smalltalk

- Type `VpcsUserIdentification`
- Code interne `userPassword`
- Nom d'utilisation `userPassword`
- get/set `userPassword` / `userPassword:`

COM

- Type `String`
- Code interne `password`
- Nom d'utilisation `password`
- get/set C++ get non disponible / `public void setPassword(String p)`

2.12.5. Nom de la machine abritant la gateway Java/Com VisualAge Pacbase

Description

Cet attribut contient l'adresse TCP-IP de la machine qui abrite le gestionnaire de communication permettant de transmettre les messages aux Composants Applicatifs élémentaires.

Cet attribut est systématiquement disponible sur le nœud racine.

Il doit être impérativement renseigné dans le cas de l'utilisation d'une gateway Java/Com VisualAge Pacbase.

Il est accessible en lecture ou en écriture.

Java

- Type `String`
- Code interne `host`
- Nom d'utilisation `Host`
- get/set
```
public String getHost()
public void setHost(String h)
```

Smalltalk

Non disponible.

COM

- Type `String`
- Code interne `host`
- Nom d'utilisation `host`
- get/set C++ `public String getHost()` / `public void setHost(String h)`

2.12.6. Port IP associé au gestionnaire de communication

Description

Cet attribut contient le port TCP-IP associé au gestionnaire de communication permettant de transmettre les messages aux Composants Applicatifs élémentaires.

Cet attribut est systématiquement disponible sur le nœud racine.

Ce port doit être identique à celui utilisé pour la gateway. Par défaut, il vaut 5647 des deux côtés.

Il est accessible en lecture ou en écriture.

Java

- Type `int`
- Code interne `port`
- Nom d'utilisation `Port`
- get/set
```
public int getPort()
public void setPort(int p)
```

Smalltalk

Non disponible.

COM

- Type `int`
- Code interne `port`
- Nom d'utilisation `port`
- get/set C++ `public int getPort()` / `public void setPort(int p)`

2.12.7. Initialisation de l'adresse du fichier des plateformes

Description

Cet attribut contient le chemin complet du fichier des plateformes utilisé par le gestionnaire de communication pour récupérer les caractéristiques du protocole de communication permettant d'accéder à un Composant Applicatif élémentaire.

Cet attribut est systématiquement disponible sur le nœud racine.

Il ne doit être utilisé que lorsque l'application Java accède au middleware en local et non via une gateway.

Il est accessible en lecture ou en écriture.

Java

Cette information n'est pas référencée en tant qu'attribut dans l'API de la Proxy. Elle est modifiable en exécutant la méthode suivante :

- get/set
  get non disponible
```
public void setLocationsFile(String f)
```

Smalltalk

Non disponible.

COM

- Type `String`
- Code interne `locationsFile`
- Nom d'utilisation `locationsFile`
- get/set C++ get non disponible / `public void setLocationsFile(String f)`

2.12.8. Choix de l'adaptateur de communication

Description

Cet attribut permet de définir le mode de communication utilisé (`MiddlewareAdapter`, `GatewayAdapter`, etc..) en indiquant le nom de la classe du `ServerAdapter` choisi ou bien, pour la cible Java, en passant une instance du `ServerAdapter` choisi.

Cet attribut est systématiquement disponible sur le nœud racine.

Java

- Type `ServerAdapterName`
- Code interne `serverAdapterName`
- Nom d'utilisation `Server Adapter Name`
- get/set
```
public String getServerAdapterName ()
public void setServerAdapterName(String className)
```
- Type `ServerAdapter`
- Code interne `serverAdapter`
- Nom d'utilisation `Server Adapter`
- get/set
```
public ServerAdapter getServerAdapter ()
public void setServerAdapter(ServerAdapter serverAdapter)
```

Smalltalk

Non disponible

COM

- Type `String`
- Code interne `serverAdapterName`
- Nom d'utilisation `serverAdapterName`
- get/set
```
public String getServerAdapterName ()
public void setServerAdapterName(String className)
```

2.12.9. Gestion des paramètres de communication

Description

Les attributs « paramètres de communication » sont lus et affectés en utilisant les méthodes **`getProperty/setProperty`**. Ces attributs définissent notamment les paramètres nécessaires pour effectuer une communication avec les serveurs en fonction du mode de communication utilisé (**`MiddlewareAdapter, GatewayAdapter`**, etc.).

La liste suivante est une liste exhaustive des paramètres de communication :

- **`locationsFile`**
- **`location`**
- **`folder`**
- **`port`**
- **`host`**
- **`traceFile`**
- **`traceLevel`**
- **`userName`**
- **`password`**
- **`pcvFlag`**
- **`anonymousKey`**
- **`securityKey`**

Cet attribut est systématiquement disponible sur le nœud racine.

Java

- Type **`Object`**
- Code interne **`property`**
- Nom d'utilisation **`Property`**
- get/set **`public Object getProperty (String attribut_name)/ public void setProperty(String attribut_name, Object attribut_value)`**

Smalltalk

Non disponible.

COM

- Type **`Int Double String`**
- Code interne **`setDoubleProperty setIntProperty setStringProperty`**
- Nom d'utilisation **`setDoubleProperty setIntProperty setStringProperty`**
- get/set get non disponible /
  `public void setIntProperty(String attribut_name, Int value)`
  `public void setDoubleProperty(String attribut_name, Double value)`
  `public void setStringProperty(String attribut_name, String value)`

2.13. Gestion des conversations asynchrones

2.13.1. Détermination du type de conversation

Description

Cet attribut est un booléen définissant le type de conversation courant du Dossier. Il doit être positionné à `true` pour la prise en compte d'une conversation de type asynchrone, à `false` pour une conversation de type synchrone. Il est initialisé à `false` par défaut.
Cet attribut est systématiquement disponible sur le nœud racine. Il est accessible en lecture / écriture.

Java

- Type `Boolean`
- Code interne `asynchronous`
- Nom d'utilisation `Asynchronous Mode`
- get/set
```
public Boolean isAsynchronous()
public void setAsynchronous(Boolean a)
```

Smalltalk

- Type `Boolean`
- Code interne `isAsynchronous:`
- Nom d'utilisation `isAsynchronous:`
- get/set `isAsynchronous` / `isAsynchronous:`

COM

- Type `Boolean`
- Code interne `asynchronous`
- Nom d'utilisation `asynchronous`
- get/set C++
```
public Boolean getAsynchronous()
public void setAsynchronous(Boolean a)
```

2.13.2. Dernier identifiant de conversation asynchrone

Description

Cet attribut contient l'identifiant de réponse de la dernière requête exécutée avec une conversation de type asynchrone sur la localisation courante.

Cet attribut est systématiquement disponible sur le nœud racine.

Il est accessible en lecture.

Java

- Type `com.ibm.vap.generic.ServerActionContext`
- Code interne `lastReplyContext`
- Nom d'utilisation `Last Reply Context`
- get/set `public ServerActionContext getLastReplyContext()`
  set non disponible

Smalltalk

- Type `VapServerActionContext`
- Code interne `lastReplyId`
- Nom d'utilisation `lastReplyId`
- get/set `lastReplyId` / set non disponible

COM

- Type `ServerActionContext`
- Code interne `lastReplyContext`
- Nom d'utilisation `lastReplyContext`
- get/set C++ `public ServerActionContext getLastReplyContext()`
  set non disponible

2.13.3. Nombre maximum de réponses en attente

Description

Cet attribut contient le nombre maximum de requêtes en attente de réponse pour la localisation courante. Ce nombre est un paramètre spécifique (`MWMAXREPLY`) des conversations asynchrones indiqué dans le fichier des plateformes.

Cet attribut est systématiquement disponible sur le nœud racine.

Il est accessible uniquement en lecture.

Java

- Type `int`
- Code interne `maximumReplyCount`
- Nom d'utilisation `Maximum Reply Count`
- get/set `public int getMaximumReplyCount()`
  set non disponible

Smalltalk

- Type `Integer`
- Code interne `maximumReplyCount`
- Nom d'utilisation `maximumReplyCount`
- get/set `maximumReplyCount` / set non disponible

COM

- Type `int`
- Code interne `maximumReplyCount`
- Nom d'utilisation `maximumReplyCount`
- get/set C++ `public int getMaximumReplyCount()`/ set non disponible

2.13.4. Nombre de réponses en attente

Description

Cet attribut contient le nombre de réponses asynchrones en attente pour le Dossier. Il est initialisé à zéro à chaque changement de localisation.

Il est incrémenté à l'exécution de toute requête utilisant un type de conversation asynchrone sauf pour celles de type mise à jour.

Il est décrémenté à chaque réception de réponse ou lorsque des réponses en attente sont annulées.

Cet attribut est systématiquement disponible sur le nœud racine.

Il est accessible uniquement en lecture.

Java

- Type `int`
- Code interne `pendingReplyCount`
- Nom d'utilisation `Pending Reply Count`
- get/set `public int getPendingReplyCount()`/ set non disponible

Smalltalk

- Type `Integer`
- Code interne `pendingReplyCount`
- Nom d'utilisation `pendingReplyCount`
- get/set `pendingReplyCount` / set non disponible

COM

- Type `int`
- Code interne `pendingReplyCount`
- Nom d'utilisation `pendingReplyCount`
- get/set C++ `public int getPendingReplyCount()`/ set non disponible

2.14. Temps de conversation

2.14.1. Temps de communication

Description

Cet attribut contient le temps total de communication de la dernière conversation avec le serveur exprimé en millisecondes.

Il est initialisé à une valeur vide.

Cet attribut est systématiquement disponible sur le nœud racine.

Il est accessible en lecture.

Java

- Type `int`
- Code interne `communicationResponseTime`
- Nom d'utilisation `Communication Response Time`
- get/set `public int getCommunicationResponseTime()`
  set non disponible

Smalltalk

- Type `Float`
- Code interne `communicationResponseTime`
- Nom d'utilisation `communicationResponseTime`
- get/set `communicationResponseTime` / set non disponible

COM

- Type `int`
- Code interne `communicationResponseTime`
- Nom d'utilisation `communicationResponseTime`
- get/set C++ `public int getCommunicationResponseTime()`
  set non disponible

2.14.2. Temps d’exécution du traitement serveur

Description

Cet attribut contient le temps total d’exécution du traitement serveur de la dernière conversation exprimé en millisecondes.

Il est initialisé à une valeur vide.

Cet attribut est systématiquement disponible sur le nœud racine.

Il est accessible en lecture.

Java

- Type `int`
- Code interne `serverResponseTime`
- Nom d’utilisation `Server Response Time`
- get/set `public int getServerResponseTime()`
  set non disponible

Smalltalk

- Type `Float`
- Code interne `serverResponseTime`
- Nom d’utilisation `serverResponseTime`
- get/set `serverResponseTime` / set non disponible

COM

- Type `int`
- Code interne `serverResponseTime`
- Nom d’utilisation `serverResponseTime`
- get/set C++ `public int getServerResponseTime()`
  set non disponible

2.15. Gestion des sous-schémas

2.15.1. Liste des sous-schémas disponibles

Description

Cet attribut contient la liste des sous-schémas disponibles sur le nœud. Les sous-schémas sont spécifiés dans la description de la Vue Logique associée au nœud.

Cet attribut est disponible si les Composants Applicatifs gèrent la présence des Rubriques (options **`VECTPRES=YES`** ou **`CHECKSER=YES`**) et si le nœud comporte au moins un sous-schéma.

Il est accessible en lecture.

Java

- Type **`SubSchema[]`**
- Code interne **`subSchemaList`**
- Nom d'utilisation **`SubSchema List`**
- get/set **`public SubSchema[] getSubSchemaList()`**
  set non disponible

Smalltalk

- Type **`VpcsOrderedCollection`** contenant **`VpcsSubSchema`**
- Code interne **`subSchemaList`**
- Nom d'utilisation **`subSchemaList`**
- get/set **`subSchemaList`** / set non disponible

COM

Disponible avec une API de parcours de collection

- Nb d'éléments **`public Int getSubSchemasCount()`**
- Elément **`public String getSubSchemasElementAt(Int i)`**

2.15.2. Sous-schéma à prendre en compte

Description

Cet attribut contient le sous-schéma à prendre en compte lors de l'exécution d'une action de sélection, lecture ou mise à jour.

Les sous-schémas sont spécifiés dans la description de la Vue Logique associée au nœud.

Cet attribut est disponible si les Composants Applicatifs gèrent la présence des Rubriques (options **`VECTPRES=YES`** ou **`CHECKSER=YES`**) et si le nœud comporte au moins un sous-schéma.

Il est accessible en lecture et en écriture.

Java

- Type **`SubSchema`**
- Code interne **`subSchema`**
- Nom d'utilisation **`Current SubSchema`**
- get/set
```
public String getSubSchema()
public void setSubSchema(SubSchema SubSchema)
```

Smalltalk

- Type **`VpcsSubSchema`**
- Code interne **`subSchema`**
- Nom d'utilisation **`subSchema`**
- get/set **`subSchema`** / **`subSchema:`**

COM

- Type **`String`**
- Code interne **`subSchema`**
- Nom d'utilisation **`subSchema`**
- get/set C++ **`String getSubSchema()`** / **`void setSubSchema(String s)`**

2.16. Utilisation d'une JTable

2.16.1. Affichage de la collection des instances dans une JTable

Description

Cet attribut ne concerne que Java.

Il vous permet d'intégrer dans une application une **`JTable`**, composant swing constitué de plusieurs lignes et de plusieurs colonnes et d'afficher la collection des instances de Vue Logique dans cette **`JTable`** par l'intermédiaire de l'attribut **`tableModel`**.

Cet attribut est disponible sur tous les types de nœud si vous avez choisi l'option de génération **`Utiliser Swing`**.

Il est accessible en lecture/écriture.

Java

- Type **`PacbaseTableModel`**
- Code interne **`tableModel`**
- Nom d'utilisation **`TableModel`**
- get/set
```
public getTableModel
public setTableModel(PacbaseTableModel)
```

Smalltalk

Non disponible

COM

Non disponible

2.16.2. Affichage des dossiers modifiés dans une JTable

Description

Cet attribut ne concerne que Java.

Il vous permet d'intégrer dans une application une **`JTable`**, composant swing constitué de plusieurs lignes et de plusieurs colonnes et d'afficher la collection des dossiers modifiées dans cette **`Jtable`** par l'intermédiaire de l'attribut **`updatedFoldersTableModel`**.

Cet attribut est disponible sur tous les nœuds de type racine si vous avez choisi l'option de génération **`Utiliser Swing`**.

Il est accessible en lecture/écriture.

Java

- Type `PacbaseUpdateTableModel`
- Code interne `updatedFoldersTableModel`
- Nom d'utilisation `UpdatedFoldersTableModel`
- get/set `public getUpdatedFoldersTableModel` `public setUpdatedFoldersTableModel(PacbaseUpdateTableModel)`

Smalltalk

Non disponible

COM

Non disponible

2.16.3. Affichage des instances modifiées dans une JTable

Description

Cet attribut ne concerne que Java.

Il vous permet d'intégrer dans une application une `JTable`, composant swing constitué de plusieurs lignes et de plusieurs colonnes et d'afficher la collection des instances modifiées dans cette `Jtable` par l'intermédiaire de l'attribut `updatedInstancesTableModel`.

Cet attribut est disponible sur tous les nœuds de type racine si vous avez choisi l'option de génération `Utiliser Swing`.

Il est accessible en lecture/écriture.

Java

- Type `PacbaseUpdateTableModel`
- Code interne `updatedInstancesTableModel`
- Nom d'utilisation `UpdatedInstancesTableModel`
- get/set `public getUpdatedInstancesTableModel` `public setUpdatedInstancesTableModel(PacbaseUpdateTableModel)`

Smalltalk

Non disponible

COM

Non disponible

2.16.4. Affichage de la collection d'instances destinées ou rendues par un Service Utilisateur dans une JTable

Description

Cet attribut ne concerne que Java.

Il vous permet d'intégrer dans une application une **`JTable`**, composant swing constitué de plusieurs lignes et de plusieurs colonnes et d'afficher la collection des instances de Vue Logique destinées ou rendues par un service utilisateur dans cette **`JTable`** par l'intermédiaire de l'attribut **`tableModel`**.

Cet attribut est disponible sur tous les types de nœud si vous avez choisi l'option de génération **`Utiliser Swing`**.

Il est accessible en lecture/écriture.

Java

- Type **`PacbaseTableModel`**
- Code interne **`tableModel`**
- Nom d'utilisation **`TableModel`**
- get/set

```
public getUserInputTableModel
public setUserInputTableModel(PacbaseTableModel)
public getUserOutputTableModel
public setUserOutputTableModel(PacbaseTableModel)
```

Smalltalk

Non disponible

COM

Non disponible

3. Actions

Une action correspond à un traitement que la Proxy Vue Logique peut exécuter. Lorsqu'une action nécessite des paramètres pour être exécutée ou lorsqu'elle rend des résultats, ils sont passés par l'intermédiaire des attributs de la Proxy Vue Logique.

Les actions d'une Proxy Vue Logique se répartissent en deux types :

- Les **actions locales** qui permettent d'effectuer une opération de mise à jour sur les instances de Vue Logique mémorisées par la Proxy Vue Logique.
- Les **actions serveur** qui permettent d'exécuter un traitement sur le serveur. Si le serveur utilise un buffer utilisateur, ce type d'action échange son contenu à chaque conversation avec le serveur.

Ces actions permettent donc de déclencher des traitements locaux internes à la Proxy Vue Logique ou des traitements distants. Ce sont les traitements standard de sélection et de mise à jour et les traitements utilisateur définis sur le Composant Applicatif associé à la Proxy Vue Logique.

Remarque : La disponibilité de chacune des actions est indiquée au niveau du paragraphe « comportement » de chaque action. Dans le cas de la cible Java, si une action est utilisée bien qu'elle soit non disponible (utilisation à tort de la méthode publique), une exception `java.lang.IllegalStateException` du Runtime sera levée.

3.1. Actions exécutées localement

3.1.1. Mises à jour

3.1.1.1. Création d'une instance de Vue Logique

Comportement

Cette action permet de créer localement une instance de Vue Logique.

Cette action est valide si :

- l'instance n'existe pas en local
- le contrôle de toutes les Rubriques de l'instance n'a pas renvoyé d'erreur
- l'instance parente d'un nœud dépendant est présente en local
- l'instance pour une même instance parente est l'unique instance présente en local pour un nœud dépendant de cardinalité maximum 1
- le Dossier est dans un état « modifiable ».

Si cette action est valide :

- le compteur du nombre total de services de mise à jour est incrémenté de 1
- le compteur du nombre de services de mise à jour associé au nœud est incrémenté de 1
- le compteur du nombre total d'instances locales est incrémenté de 1
- le container de la liste d'instances associé au nœud intègre la nouvelle instance
- l'attribut de présentation des Dossiers modifiés intègre la nouvelle modification

- émission de l'événement de non détection d'erreur.

Si cette action est non valide :

- ajout de l'erreur dans l'objet erreur
- émission de l'événement d'erreur locale.

Cette action est disponible si le Composant Applicatif permet les mises à jour sur la Vue Logique et si tous les nœuds dépendants de cardinalité minimum 1 sont présents dans la Vue de Dossier.

Java

- Signature `public void createInstance() throws LocalException`
- Nom d'utilisation `Create Instance`

Smalltalk

- Signature `createInstance`
- Nom d'utilisation `createInstance`

COM

- Signature C++ `public void createInstance()`
- Nom d'utilisation `createInstance`

3.1.1.2. Modification d'une instance de Vue Logique

Comportement

Cette action permet de modifier localement une instance de Vue Logique.
Cette action est valide si :

- l'instance existe dans l'attribut de présentation d'une instance,
- l'instance existe localement,
- le contrôle de toutes les Rubriques de l'instance n'a pas renvoyé d'erreur,
- le Dossier est dans un état « modifiable ».

Si cette action est valide :

- le compteur du nombre total de services de mise à jour est incrémenté de 1 si aucun mouvement de mise à jour n'est déjà associé à cette instance,
- le compteur du nombre de services de mise à jour associé au nœud est incrémenté de 1 si aucun mouvement de mise à jour n'est déjà associé à cette instance,
- le container de la liste d'instances associé au nœud intègre la modification,
- l'attribut de présentation des Dossiers modifiés intègre la nouvelle modification,
- émission de l'événement de non détection d'erreur.

Si cette action est non valide :

- ajout de l'erreur dans l'objet erreur,
- émission de l'événement d'erreur locale.

Cette action est disponible si le Composant Applicatif permet les mises à jour sur la Vue Logique.

Java

- Signature `public void modifyInstance() throws LocalException`
- Nom d'utilisation `Modify Instance`

Smalltalk

- Signature `modifyInstance`
- Nom d'utilisation `modifyInstance`

COM

- Signature C++ `public void modifyInstance()`
- Nom d'utilisation `modifyInstance`

3.1.1.3. Suppression d'une instance de Vue Logique

Comportement

Cette action permet de supprimer localement une instance de Vue Logique. Elle supprime localement en cascade toutes les instances des nœuds dépendants.

Cette action est valide si :

- l'instance existe localement
- l'instance existe dans l'attribut de présentation d'une instance
- l'instance parente d'un nœud dépendant est présente en local
- le Dossier est dans un état « modifiable »

Si cette action est valide :

- le compteur du nombre total de services de mise à jour est incrémenté de 1 si aucun mouvement de mise à jour n'est déjà associé à cette instance
- le compteur du nombre de services de mise à jour associé au nœud est incrémenté de 1 si aucun mouvement de mise à jour n'est déjà associé à cette instance
- le compteur du nombre total d'instances locales est décrémenté du nombre d'instances dépendantes implicitement supprimées + 1.
- l'instance est supprimée du container de la liste d'instances associé au nœud
- l'attribut de présentation des Dossiers modifiés intègre la nouvelle modification
- toutes les instances locales qui dépendent de l'instance supprimée sont supprimées
- émission de l'événement de non détection d'erreur

Si cette action est non valide :

- ajout de l'erreur dans l'objet erreur
- émission de l'événement d'erreur locale

Cette action est disponible si le Composant Applicatif permet les mises à jour sur la Vue Logique.

Java

- Signature `public void deleteInstance() throws LocalException`
- Nom d'utilisation `Delete Instance`

Smalltalk

- Signature `deleteInstance`
- Nom d'utilisation `deleteInstance`

COM

- Signature C++ `public void deleteInstance()`
- Nom d'utilisation `deleteInstance`

3.1.2. Annulation des mises à jour

3.1.2.1. Annulation des mises à jour d'un Dossier

Comportement

Cette action permet d'annuler toutes les mises à jour locales effectuées sur une instance de Dossier tous nœuds confondus à partir de la première mise à jour locale.

Cette action est valide si :

- l'instance existe dans l'attribut de présentation d'une instance du nœud racine.

Si cette action est valide :

- l'image initiale de l'instance et des instances dépendantes est restaurée dans le cache local, dans les attributs de présentation et dans les containers de liste d'instances
- le compteur du nombre total de services de mise à jour est recalculé
- le compteur du nombre de services de mise à jour associé au nœud est recalculé
- le compteur du nombre total d'instances locales est recalculé
- l'instance est supprimée de l'attribut de présentation des Dossiers modifiés
- émission de l'événement de non détection d'erreur.

Si cette action est non valide :

- ajout de l'erreur dans l'objet erreur
- émission de l'événement d'erreur locale.

Cette action est disponible si au moins un des Composants Applicatifs du Dossier gère une Vue Logique en mise à jour.

Java

- Signature `public void undoLocalFolderUpdates({Nom de Classe générée} DataUpdate d)  throws LocalException`
- Nom d'utilisation `Undo Local Folder Updates`

Smalltalk

- Signature `undoLocalFolderUpdatesFor: UpdatedDataDescription{CodeVueLogique}{suffixe}`
- Nom d'utilisation `undoLocalFolderUpdatesFor:`

COM

- Signature C++ `public void undoLocalFolderUpdates({Nom de Classe générée} DataUpdate d)`
- Nom d'utilisation `undoLocalFolderUpdates({Nom de Classe générée} DataUpdate d)`

3.1.2.2. Annulation des mises à jour de tous les Dossiers

Comportement

Cette action permet d'annuler toutes les mises à jour locales effectuées sur toutes les instances de Dossier tous nœuds confondus.

Après l'action :

- les images initiales des instances de tous les Dossiers et de toutes leurs instances dépendantes sont restaurées dans le cache local, dans les attributs de présentation et dans les containers de liste d'instances
- le compteur du nombre total de services de mise à jour est recalculé
- le compteur du nombre de services de mise à jour associé au nœud est recalculé
- le compteur du nombre total d'instances locales est recalculé
- l'attribut de présentation des Dossiers modifiés est réinitialisé
- émission de l'événement de non détection d'erreur

Si cette action est non valide :

- ajout de l'erreur dans l'objet erreur
- émission de l'événement d'erreur locale

Cette action est disponible si au moins un des Composants Applicatifs du Dossier gère une Vue Logique en mise à jour.

Java

- Signature `public void undoAllLocalFolderUpdates()`
- Nom d'utilisation `Undo all local folder updates`

Smalltalk

- Signature `undoAllLocalFolderUpdates`
- Nom d'utilisation `undoAllLocalFolderUpdates`

COM

- Signature C++ `public void undoAllLocalFolderUpdates()`
- Nom d'utilisation `undoAllLocalFolderUpdates`

3.1.2.3. Annulation des mises à jour d'une instance de noeud

Comportement

Cette action permet d'annuler toutes les mises à jour locales effectuées sur une instance du nœud concerné à partir de la première mise à jour locale. Cette action prend comme paramètre une instance du nœud concerné.

Cette action est valide si :

- l'instance passée en paramètre est une instance du nœud qui a été mise à jour localement.

Si cette action est valide :

- l'image initiale de l'instance et des instances dépendantes (dans le cas où le statut de modification de l'instance du nœud conerné n'est pas à l'état #Modified) est restaurée dans le cache local, dans les attributs de présentation et dans les containers de liste d'instances
- le compteur du nombre total de services de mise à jour est recalculé
- le compteur du nombre de services de mise à jour associé au nœud est recalculé
- le compteur du nombre total d'instances locales est recalculé
- l'instance et toutes les instances dépendantes sont supprimées de leur attribut respectif de présentation des instances modifiées
- l'attribut de présentation des Dossiers est mis à jour
- émission de l'événement de non détection d'erreur.

Si cette action est non valide :

- ajout de l'erreur dans l'objet erreur
- émission de l'événement d'erreur locale.

Cette action est disponible sur un nœud racine ou dépendant d'une Proxy Vue de Dossier lorsque le Composant Applicatif associé au nœud comporte un service de mise à jour.

Java

- Signature
```
public void undoLocalUpdate({Nom de Classe générée}
DataUpdate d) throws LocalException
```
- Nom d'utilisation `Undo Local Update`

Smalltalk

- Signature `undoLocalUpdateFor: aDataDescription`
- Nom d'utilisation `undoLocalUpdateFor:`

COM

- Signature C++
```
public void undoLocalUpdate({Nom de Classe générée}
DataUpdate d)
```
- Nom d'utilisation `undoLocalUpdate({Nom de Classe générée} DataUpdate d)`

3.1.2.4. Annulation des mises à jour de toutes les instances d'un noeud

Comportement

Cette action permet d'annuler toutes les mises à jour locales effectuées sur un nœud de toutes les instances de la hiérarchie courante à partir de la première mise à jour locale.

Après l'action :

- les images initiales des instances du noeud pour la hiérarchie courante et de toutes leurs instances dépendantes (dans le cas où le statut de modification d'une instance du nœud concerné n'est pas à l'état #Modified) sont restaurées dans le cache local, dans les attributs de présentation et dans les containers de liste d'instances
- le compteur du nombre total de services de mise à jour est recalculé
- le compteur du nombre de services de mise à jour associé au nœud est recalculé
- le compteur du nombre total d'instances locales est recalculé
- les instances et toutes leurs instances dépendantes sont supprimées de leur attribut respectif de présentation des instances modifiées
- l'attribut de présentation des Dossiers est mis à jour
- émission de l'événement de non détection d'erreur

Cette action est disponible sur un nœud racine ou dépendant d'une Proxy Vue de Dossier lorsque le Composant Applicatif associé au nœud comporte un service de mise à jour.

Java

- Signature `public void undoAllLocalUpdate()`
- Nom d'utilisation `Undo all local update`

Smalltalk

- Signature `undoAllLocalUpdate`
- Nom d'utilisation `undoAllLocalUpdate`

COM

- Signature C++ `public void undoAllLocalUpdate()`
- Nom d'utilisation `undoAllLocalUpdate`

3.1.3. Gestion des Services Utilisateur

3.1.3.1. Affectation d'une instance à un Service Utilisateur

Comportement

Cette action permet de créer localement sur un nœud une nouvelle instance de Vue Logique réservée pour l'exécution du prochain Service Utilisateur.

Cette action est valide si :

- l'instance existe dans l'attribut de présentation d'une instance liée à un Service Utilisateur.

Si cette action est valide :

- le compteur du nombre d'instances de Vue Logique réservé à un Service Utilisateur est incrémenté de 1
- l'attribut de présentation d'instances destinées à un Service Utilisateur intègre l'instance
- émission de l'événement de non détection d'erreur.

Si cette action est non valide :

- ajout de l'erreur dans l'objet erreur
- émission de l'événement d'erreur locale.

Cette action est disponible si le Composant Applicatif associé au nœud gère au moins un Service Utilisateur.

Java

- Signature `public void createUserInstance()throws LocalException`
- Nom d'utilisation `Create User Instance`

Smalltalk

- Signature `createUserServiceInstance`
- Nom d'utilisation `createUserServiceInstance`

COM

- Signature C++ `public void createUserInstance()`
- Nom d'utilisation `createUserInstance`

3.1.3.2. Modification d'une instance affectée

Comportement

Cette action permet de modifier localement sur un nœud une instance de Vue Logique réservée pour l'exécution du prochain Service Utilisateur.

Cette action est valide si :

- l'instance existe dans l'attribut de présentation d'instances réservées à un Service Utilisateur
- l'instance existe dans l'attribut de présentation d'une instance liée à un Service Utilisateur.

Si cette action est valide :

- l'attribut de présentation d'instances destinées à un Service Utilisateur intègre la modification
- émission de l'événement de non détection d'erreur.

Si cette action est non valide :

- ajout de l'erreur dans l'objet erreur
- émission de l'événement d'erreur locale.

Cette action est disponible si le Composant Applicatif associé au nœud gère au moins un Service Utilisateur.

Java

- Signature `public void modifyUserInstance() throws LocalException`
- Nom d'utilisation `Modify User Instance`

Smalltalk

- Signature `modifyUserServiceInstance`
- Nom d'utilisation `modifyUserServiceInstance`

COM

- Signature C++ `public void modifyUserInstance()`
- Nom d'utilisation `modifyUserInstance`

3.1.3.3. Suppression d'une instance affectée

Comportement

Cette action permet de supprimer localement sur un nœud une instance de Vue Logique réservée pour l'exécution du prochain Service Utilisateur.

Cette action est valide si :

- l'instance existe dans l'attribut de présentation d'instances réservées à un Service Utilisateur
- l'instance existe dans l'attribut de présentation d'une instance liée à un Service Utilisateur

Si cette action est valide :

- le compteur du nombre d'instances de Vue Logique réservé à un Service Utilisateur est décrémenté de 1
- l'attribut de présentation d'instances destinées à un Service Utilisateur intègre la suppression de l'instance
- émission de l'événement de non détection d'erreur.

Si cette action est non valide :

- ajout de l'erreur dans l'objet erreur
- émission de l'événement d'erreur locale

Cette action est disponible si le Composant Applicatif associé au nœud gère au moins un Service Utilisateur.

Java

- Signature `public void deleteUserInstance() throws LocalException`
- Nom d'utilisation `Delete User Instance`

Smalltalk

- Signature `deleteUserServiceInstance`
- Nom d'utilisation `deleteUserServiceInstance`

COM

- Signature C++ `public void deleteUserInstance()`
- Nom d'utilisation `deleteUserInstance`

3.1.4. Navigation locale dans les Dossiers

Définition de la **Règle globale d'alimentation des attributs de présentation des instances** :

Lorsque l'attribut de présentation d'une instance d'un nœud père contient une instance valide, les attributs de présentation d'une instance et de liste d'instances de ses nœuds dépendants sont alimentés en fonction des règles suivantes :

- si le nœud dépendant a une cardinalité maximale de n, son attribut de présentation des instances est alimenté par toutes les instances contenues dans le cache local qui dépendent de l'instance courante du nœud père. S'il n'y a qu'une instance dans le cache local, son attribut de présentation d'une instance est aussi alimenté par cette instance.
- si le nœud a une cardinalité maximale de 1, son attribut de présentation d'instance est alimenté par l'instance qui dépend de l'instance courante du nœud père si elle est présente dans le cache local.
- si le nœud ne répond pas aux deux règles ci-dessus, son attribut de présentation d'instance et son attribut de présentation de liste sont alimentés avec une valeur vide.

3.1.4.1. Sélection courante d'une instance dans un dossier

Comportement

Cette action permet d'affecter à l'attribut **présentation d'une instance** d'un nœud une instance du même type, en particulier une instance provenant de l'attribut **présentation d'une liste d'instances**.

Cette action est valide si :

- le paramètre fourni en entrée de cette action est une instance de `DataDescription.`

Si cette action est valide :

- l'attribut de présentation d'une instance contient l'instance à affecter
- les attributs de présentation d'une instance et de liste d'instances des nœuds dépendants sont alimentés suivant la règle globale d'alimentation

- l'identifiant de verrouillage du Dossier est alimenté si l'instance à affecter est une instance d'un nœud racine ayant été précédemment verrouillée
- émission de l'événement de non détection d'erreur.

Si cette action est non valide :

- ajout de l'erreur dans l'objet erreur
- émission de l'événement d'erreur locale.

Cette action est systématiquement disponible.

Java

- Signature `public void getDetailFromDataDescription({Nom de classe générée}Data d) throws LocalException`
- Nom d'utilisation `Get Detail From Data Description`

Smalltalk

- Signature `getDetailFromDataDescription: aDataDescription`
- Nom d'utilisation `getDetailFromDataDescription:`

COM

- Signature C++ `public void getDetailFromData({Nom de classe générée}Data d)`
- Nom d'utilisation `getDetailFromData({Nom de classe générée}Data d)`

3.1.4.2. Sélection d'une instance associée à un Service Utilisateur

Comportement

Cette action permet d'affecter à l'attribut **présentation d'une instance destinée à un Service Utilisateur** d'un nœud une instance du même type, en particulier une instance provenant de l'attribut **présentation d'une liste d'instance destinée à un Service Utilisateur**.

Après cette action :

- l'attribut de présentation d'une instance destinée à un Service Utilisateur contient l'instance à affecter
- émission de l'événement de non détection d'erreur.

Si cette action est non valide :

- ajout de l'erreur dans l'objet erreur
- émission de l'événement d'erreur locale.

Cette action est disponible lorsque le Composant Applicatif associé au nœud a au moins un Service Utilisateur.

Java

- Signature `public void getUserDetailFromDataDescription({Nom de classe générée}Data d) throws LocalException`
- Nom d'utilisation `Get User Detail From Data Description`

Smalltalk

- Signature `getUserDetailFromDataDescription:aDataDescription`
- Nom d'utilisation `getUserDetailFromDataDescription:`

COM

- Signature C++ `public void getUserDetailFromData({Nom de classe générée}Data d)`
- Nom d'utilisation `getUserDetailFromData({Nom de classe générée}Data d)`

3.1.5. Initialisations diverses

3.1.5.1. Initialisation de la collection

Comportement

Cette action permet d'éliminer du cache local toutes les instances du nœud et de ses dépendants.

Après cette action :

- l'attribut de présentation d'une instance du nœud est initialisé à vide
- l'attribut de présentation de listes d'instance du nœud est initialisé à vide
- l'attribut de présentation d'une instance des nœuds dépendants est initialisé à vide
- l'attribut de présentation de listes d'instance des nœuds dépendants est initialisé à vide

Cette action est systématiquement disponible pour tous les nœuds.

Java

- Signature `public void resetCollection ()`
- Nom d'utilisation `resetCollection`

Smalltalk

- Signature `resetCollection`
- Nom d'utilisation `resetCollection`

COM

- Signature C++ `public void resetCollection()`
- Nom d'utilisation `resetCollection`

3.1.5.2. Initialisation des méthodes d'extraction

Comportement

Cette action permet d'initialiser l'attribut **méthode d'extraction à exécuter** du noeud et de tous ses dépendants avec une valeur vide.

Après cette action :

- l'attribut méthode d'extraction à exécuter de chaque nœud concerné contient une valeur vide

Cette action est systématiquement disponible pour tous les nœuds.

Java

- Signature `public void resetExtractMethodCodes()`
- Nom d'utilisation `Reset Extract Method Codes`

Smalltalk

- Signature `resetExtractMethodCodes`
- Nom d'utilisation `resetExtractMethodCodes`

COM

- Signature C++ `public void resetExtractMethodCodes()`
- Nom d'utilisation `resetExtractMethodCodes`

3.1.5.3. Initialisation des Services Utilisateur

Comportement

Cette action permet d'initialiser l'attribut **Service Utilisateur à exécuter** du nœud et de tous ses dépendants avec une valeur vide.

Après cette action :

- l'attribut Service Utilisateur à exécuter de chaque nœud concerné contient une valeur vide

Cette action est disponible lorsque le Composant Applicatif associé au nœud a au moins un Service Utilisateur.

Java

- Signature `public void resetUserServiceCodes()`
- Nom d'utilisation `Reset User Service Codes`

Smalltalk

- Signature `resetUserServiceCodes`
- Nom d'utilisation `resetUserServiceCodes`

COM

- Signature C++ `public void resetUserServiceCodes()`
- Nom d'utilisation `resetUserServiceCodes`

3.1.5.4. Initialisation du container présentation d'instances destinées à un Service Utilisateur

Comportement

Cette action permet d'initialiser l'attribut **présentation d'instances destinées à un Service Utilisateur** du nœud et de tous ses dépendants avec une valeur vide.

Après cette action :

- l'attribut présentation d'instances destinées à un Service Utilisateur à exécuter de chaque nœud concerné contient une valeur vide

Cette action est disponible lorsque le Composant Applicatif associé au nœud a au moins un Service Utilisateur.

Java

- Signature `public void resetUserRows()`
- Nom d'utilisation `Reset User Rows`

Smalltalk

- Signature `resetUserServiceInputInstances`
- Nom d'utilisation `resetUserServiceInputInstances`

COM

- Signature C++ `public void resetUserRows()`
- Nom d'utilisation `resetUserRows`

3.1.5.5. Initialisation de l'option de rafraîchissement des mises à jour

Comportement

Cette action permet d'inhiber l'option de rafraîchissement des mises à jour sur le nœud ainsi que celle de ses nœuds dépendants.

Cette action est disponible lorsque le Composant Applicatif associé au nœud utilise la Vue Logique en mise à jour.

Java

- Signature `public void resetAllRefreshOption()`
- Nom d'utilisation `Reset All Refresh Option`

Smalltalk

- Signature `resetAllRefreshOption`
- Nom d'utilisation `resetAllRefreshOption`

COM

- Signature C++ `public void resetAllRefreshOption()`
- Nom d'utilisation `resetAllRefreshOption`

3.1.5.6. Initialisation des critères de sélection

Comportement

Cette action permet d'initialiser l'attribut Critères de sélection du nœud et de tous ses dépendants avec une valeur vide.

Après cette action, l'attribut Critères de sélection de chaque nœud concerné contient une valeur vide.

Cette action est systématiquement disponible pour tous les nœuds racine et dépendants.

Java

- Signature `public void resetSelectionCriterias()`
- Nom d'utilisation `Reset Selection Criterias`

Smalltalk

- Signature `resetSelectionCriterias`
- Nom d'utilisation `resetSelectionCriterias`

COM

- Signature C++ `public void resetSelectionCriterias()`
- Nom d'utilisation `resetSelectionCriterias`

3.1.5.7. Peuplement du cache local sans accès serveur

Comportement

Cette action permet de stocker une instance de Vue Logique dans le cache local sans qu'elle soit ramenée par un accès serveur et qu'elle n'ait pas le statut créé localement.

Cette action est valide si l'instance n'existe pas en local, quelque soit son statut.

Si cette action est valide :
- le compteur du nombre total d'instances locales est incrémenté de 1
- le container de la liste d'instances associé au noeud intègre la nouvelle instance
- émission de l'événement de non détection d'erreur.

Si cette action est non valide :
- ajout de l'erreur dans l'objet d'erreur
- émission de l'événement d'erreur locale

Cette action est systématiquement disponible pour tous les nœuds.

Java

- Signature `public void initializeInstance() throws LocalException`
- Nom d'utilisation `initializeInstance`

Smalltalk

Non disponible

COM

Non disponible

3.1.6. Gestion des instances référencées

3.1.6.1. Affectation d'une instance référencée

Comportement

Cette action permet d'affecter les Rubriques de type identifiant de l'instance d'un nœud référence passée en paramètre de cette action aux Rubriques de type 'foreign key' de l'instance du nœud référençant.

Cette action est valide si :
- une instance existe dans l'attribut de présentation d'une instance du nœud référençant
- l'instance du nœud référence ne contient pas une valeur vide

Si cette action est valide :
- les Rubriques de type 'foreign key' dans l'attribut de présentation d'une instance du nœud référençant sont initialisées avec les Rubriques de type identifiant du nœud référence

Si cette action est non valide :
- ajout de l'erreur dans l'objet erreur
- émission de l'événement d'erreur locale

Cette action est systématiquement disponible sur les nœuds référence

Java

- Signature `public void transferReferenceFromSelectedRow({Nom de classe générée}Data d) throws LocalException`
- Nom d'utilisation `Transfer Reference From Selected Row`

Smalltalk

- Signature `transferReferenceFromSelectedRow: aRow`
- Nom d'utilisation `transferReferenceFromSelectedRow:`

COM

- Signature C++ `public void transferReferenceFromSelectedRow({Nom de classe générée}Data d)`
- Nom d'utilisation `transferReferenceFromSelectedRow({Nom de classe générée}Data d)`

3.1.7. Gestion de l'acquisition des collections

3.1.7.1. Récupération de l'événement associé à la dernière sélection serveur

Comportement

Cette action permet de renvoyer le dernier événement émis sur un nœud lors de la dernière action de sélection.

Après cette action, l'événement est émis (présence ou pas d'une page précédente, présence ou pas d'une page suivante, lecture d'un enregistrement non trouvé, pas de lecture effectuée).

Cette action est systématiquement disponible pour tous les nœuds racine et dépendants.

Java

Non utile dans le contexte de l'utilisation de ServerEventStack.

Smalltalk

- Signature `getLastSelectResponseStatus`
- Nom d'utilisation `getLastSelectResponseStatus`

COM

Non utile dans le contexte de l'utilisation de la méthode `popServerEvent`.

3.1.8. Récupération des contextes de génération des Proxies

3.1.8.1. Contexte de génération d'un Dossier

Comportement

Cette action permet de récupérer les constantes VisualAge Pacbase du Gestionnaire de Services associé au nœud racine, sous la forme d'une collection de chaînes de caractères contenant les informations suivantes :

- nom externe du Gestionnaire de Services
- code VisualAge Pacbase du Dossier (ou du Composant Applicatif)
- code Base du Référentiel VisualAge Pacbase
- code Bibliothèque
- numéro de session de génération
- code utilisateur
- date de génération
- heure de génération
- code de la Vue de Dossier

Cette action est systématiquement disponible pour un nœud racine.

Java

- Signature `public String[] getFolderConstants()`
- Nom d'utilisation `Get Folder Constants`

Smalltalk

- Signature `getFolderConstants`
- Nom d'utilisation `getFolderConstants`

COM

Disponible avec une API de parcours de collection.

- Nb d'éléments `public Int getFolderConstantsCount()`
- Elément `public String getFolderConstantsElementAt(Int i)`

3.1.8.2. Contexte de génération d'un nœud

Comportement

Cette action permet de récupérer les constantes VisualAge Pacbase du Composant Applicatif associé au nœud.

Après cette action :

- l'action renvoie une collection de chaînes de caractères contenant les informations suivantes :
  - nom externe du Composant Applicatif
  - code VisualAge Pacbase du Composant Applicatif
  - code Base du Référentiel VisualAge Pacbase
  - code Bibliothèque
  - numéro de session de génération
  - code utilisateur
  - date de génération
  - heure de génération
  - version d'exploitation du Composant Applicatif

Cette action est systématiquement disponible pour tous les nœuds (racine, dépendants et référence).

Java

- Signature `public String[] getNodeConstants()`
- Nom d'utilisation `Get Node Constants`

Smalltalk

- Signature `getNodeConstants`
- Nom d'utilisation `getNodeConstants`

COM

Disponible avec une API de parcours de collection.

- Nb d'éléments `public Int getNodeConstantsCount()`
- Elément `public String getNodeConstantsElementAt(Int i)`

3.1.9. Gestion des sous-schémas

3.1.9.1. Aucune sélection de sous-schéma

Comportement

Cette action permet de réinitialiser l'attribut `subSchema` à vide, c'est-à-dire à ne sélectionner aucun sous-schéma.

Cette action est disponible si les Composants Applicatifs gèrent la présence des Rubriques (`VECTPRES=YES` ou `CHECKSER=YES`) et si le nœud comporte au moins un sous-schéma.

Java

- Signature `public void resetSubSchema()`
- Nom d'utilisation `Reset SubSchema`

Smalltalk

- Signature `resetSubSchema`
- Nom d'utilisation `resetSubSchema`

COM

- Signature C++ `public resetSubSchema()`
- Nom d'utilisation `resetSubSchema`

3.1.9.2. Appartenance de la Rubrique au sous-schéma

Comportement

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

Cette action est disponible si les Composants Applicatifs gèrent la présence des Rubriques (**`VECTPRES=YES`** ou **`CHECKSER=YES`**) et si le nœud comporte au moins un sous-schéma.

Java

- Signature **`public boolean belongsToSubSchema(int indexRubrique)`**
- Nom d'utilisation **`Belongs to current subschema`**

Smalltalk

- Signature **`belongsToSubschema: <#codeRubrique>`**
- Nom d'utilisation **`belongsToSubschema:`**

COM

- Signature **`belongsToSubSchema(int indexRubrique)`**
- Nom d'utilisation **`belongsToSubSchema`**

3.2. Actions exécutées sur un serveur distant

Rappel de la **Règle globale d'alimentation des attributs de présentation des instances** :

Lorsque l'attribut de présentation d'une instance d'un nœud père contient une instance valide, les attributs de présentation d'une instance et de liste d'instances de ses nœuds dépendants sont alimentés en fonction des règles suivantes :

- si le nœud dépendant a une cardinalité maximale de n, son attribut de présentation des instances est alimenté par toutes les instances contenues dans le cache local qui dépendent de l'instance courante du nœud père. S'il n'y a qu'une instance dans le cache local, son attribut de présentation d'une instance est aussi alimenté par cette instance.
- si le nœud a une cardinalité maximale de 1, son attribut de présentation d'instance est alimenté par l'instance qui dépend de l'instance courante du nœud père si elle est présente dans le cache local.
- si le nœud ne répond pas aux deux règles ci-dessus, son attribut de présentation d'instance et son attribut d'une liste d'instances sont alimentés avec une valeur vide.

3.2.1. Sélection sur un nœud

1.1.1.1. Sélection d'un ensemble d'instances

Comportement

Cette action permet de définir une collection d'instances de Vue Logique associée au nœud et de récupérer la totalité ou la première page des instances de cette collection.

Si cette action est valide :

- l'attribut de présentation d'une liste d'instances est modifié selon la valeur de l'attribut de mode de gestion de la collection
- le compteur du nombre total d'instances locales est initialisé
- le libellé et la clé du message de retour de sélection sont initialisés si la dernière instance de la collection a été récupérée
- les attributs de présentation d'une instance et de liste d'instances des nœuds dépendants sont initialisés avec une valeur vide
- émission d'un événement de non détection d'erreur
- émission de l'événement présence d'instances locales en mise à jour, en mode de gestion de collection automatique, si des instances en mises à jour sont toujours présentes dans le cache local
- émission de l'événement de récupération de la première page d'une collection si le Dossier travaille en mode `non extend` et en mode de gestion de collection automatique
- émission de l'événement de présence d'au moins une page suivante si la dernière instance de la collection n'a pas été récupérée
- émission de l'événement de récupération de la dernière page si la dernière instance de la collection a été récupérée

Si cette action est non valide :

- ajout de l’erreur dans l’objet erreur
- émission d’un événement d’erreur en fonction du type de cette erreur

Dans tous les cas :

- les compteurs de temps de conversation sont initialisés
- les attributs d’informations contextuelles – s'ils sont présents – sont initialisés

Cette action est systématiquement disponible sur les nœuds racine et références.

Java

- Signature `public void selectInstances() throws ServerException, CommunicationError, SystemError, LocalException`
- Nom d’utilisation `Select Instances`

Smalltalk

- Signature `selectInstances`
- Nom d’utilisation `selectInstances`

COM

- Signature C++ `public void selectInstances()`
- Nom d’utilisation `selectInstances`

3.2.1.2. Lecture d’une instance avec ou sans verrouillage logique

Comportement

Cette action permet de récupérer une instance de Vue Logique associée au nœud et éventuellement de se l’approprier en mise à jour exclusive.

Cette action est valide si :

- l’instance n’est pas déjà verrouillée dans le cas d’une action avec verrouillage

Si cette action est valide :

- l’attribut de présentation d’une instance est initialisé
- l’attribut de présentation de listes d’instance est modifié selon la valeur de l’attribut de mode de gestion de la collection
- le compteur du nombre total d’instances locales est initialisé
- le libellé et la clé du message de retour de sélection sont initialisés si l’instance n’a pas été récupérée
- les attributs de présentation d’une instance et de liste d’instance des nœuds dépendants sont initialisés avec une valeur vide
- émission d’un événement de non détection d’erreur
- émission de l’événement présence d’instances locales en mise à jour, en mode de gestion de collection automatique, si des instances en mises à jour sont toujours présentes dans le cache local
- l’identifiant de verrouillage de Dossier est initialisé dans le cas de demande de verrouillage

Si cette action est non valide :

- ajout de l’erreur dans l’objet erreur
- émission d’un événement d’erreur en fonction du type de cette erreur
- l’identifiant de verrouillage de Dossier est initialisé avec une valeur vide dans le cas de demande de verrouillage et le Dossier passe en état ‘non-modifiable’

Dans tous les cas :

- les compteurs de temps de conversation sont initialisés
- les attributs d'informations contextuelles – s'ils sont présents – sont initialisés

Cette action est systématiquement disponible sur tous les nœuds.

Java

- Signature
```
public void readInstance() throws LocalException,
ServerException, CommunicationError, SystemError
public void readInstanceAndLock() throws LocalException,
ServerException, CommunicationError, SystemError
```
- Nom d'utilisation
```
Read Instance
Read Instance And Lock
```

Smalltalk

- Signature `readInstance[AndLock]`
- Nom d'utilisation `readInstance[AndLock]`

COM

- Signature C++
```
public void readInstance()
public void readInstanceAndLock()
```
- Nom d'utilisation
```
readInstance
readInstanceAndLock
```

3.2.2. Sélection simultanée sur plusieurs nœud avec ou sans verrouillage

3.2.2.1. Lecture d'une instance et de sa hiérarchie immédiate

Comportement

Cette action permet de récupérer une instance de Vue Logique associée au nœud, éventuellement de se l'approprier en mise à jour exclusive, et de récupérer tout ou partie des instances des nœuds dépendants de premier niveau.

Cette action est valide si :

- l'instance n'est pas déjà verrouillée dans le cas d'une action avec verrouillage

Si cette action est valide :

- l'attribut de présentation d'une instance est initialisé avec le résultat de la sélection
- l'attribut de présentation de listes d'instance est modifié selon la valeur de l'attribut de mode de gestion de la collection
- le compteur du nombre total d'instances locales est initialisé
- le libellé et la clé du message de retour de sélection sont initialisés pour chaque nœud dépendant de premier niveau si la dernière instance de la collection a été récupérée
- les attributs de présentation d'une instance et de liste d'instances des nœuds dépendants du premier niveau de la hiérarchie sont initialisés par le résultat de la sélection sauf ceux dont le nombre d'instances échangées à été positionné à zéro qui sont initialisés avec une valeur vide. Pour l'attribut de la liste d'instances, la modification est effectuée selon la valeur de l'attribut de mode de gestion de la collection associé à chaque noeud

- les attributs de présentation d'une instance et de liste d'instances des nœuds dépendants de niveau hiérarchique supérieur à un sont initialisés à une valeur vide
- émission d'un événement de non détection d'erreur
- émission de l'événement présence d'instances locales en mise à jour, en mode de gestion de collection automatique, si des instances en mises à jour sont toujours présentes dans le cache local
- émission d'un événement d'enregistrement non trouvé sur chaque noeud qui participe à la sélection et de cardinalité maximale 1 dont l'instance n'a pas été récupérée
- l'identifiant de verrouillage de Dossier est initialisé dans le cas de demande de verrouillage

Si cette action est non valide :

- ajout de l'erreur dans l'objet erreur
- émission d'un événement d'erreur en fonction du type de cette erreur
- l'identifiant de verrouillage de Dossier est initialisé avec une valeur vide dans le cas de demande de verrouillage et le Dossier passe en état 'non-modifiable'

Dans tous les cas :

- les compteurs de temps de conversation sont initialisés
- les attributs d'informations contextuelles – s'ils sont présents – sont initialisés

Cette action est systématiquement disponible sur tous les nœuds racine et dépendants.

Java

- Signature
```
public void readInstanceWithFirstChildren() throws
LocalException, ServerException, CommunicationError,
SystemError
public void readInstanceWithFirstChildrenAndLock() throws
LocalException, ServerException, CommunicationError,
SystemError
```
- Nom d'utilisation
```
Read Instance With First Children
Read Instance With First Children And Lock
```

Smalltalk

- Signature `readInstanceWithFirstChildren[AndLock]`
- Nom d'utilisation `readInstanceWithFirstChildren[AndLock]`

COM

- Signature C++
```
public void readWithFirstChildren()
public void readWithFirstChildrenAndLock()
```
- Nom d'utilisation
```
readWithFirstChildren
readWithFirstChildrenAndLock
```

3.2.2.2. Lecture d'une instance et de sa hiérarchie complète

Comportement

Cette action permet de récupérer une instance de Vue Logique associée au nœud racine, éventuellement de se l'approprier en mise à jour exclusive, et de récupérer toutes les instances de chaque nœud dépendant quelle que soit la profondeur hiérarchique.

Cette action est valide si :

- l'instance n'est pas déjà verrouillée dans le cas d'une action avec verrouillage

Si cette action'est valide :

- l'attribut de présentation d'une instance du nœud racine est initialisé par le résultat de la sélection
- l'attribut de présentation de listes d'instance du nœud racine est modifié selon la valeur de l'attribut de mode de gestion de la collection
- le compteur du nombre total d'instances locales est initialisé
- le libellé et la clé du message de retour de sélection sont initialisés pour chaque nœud dépendant si la dernière instance de la collection a été récupérée
- les attributs de présentation d'une instance et de liste d'instances d'un nœud dépendant sont alimentés suivant la règle globale d'alimentation. Pour l'attribut de la liste d'instances, la modification est effectuée selon la valeur de l'attribut de mode de gestion de la collection associé à chaque noeud
- émission d'un événement de non détection d'erreur
- émission de l'événement présence d'instances locales en mise à jour, en mode de gestion de collection automatique, si des instances en mises à jour sont toujours présentes dans le cache local
- émission d'un événement d'enregistrement non trouvé sur chaque nœud qui participe à la sélection et de cardinalité maximale 1 dont l'instance n'a pas été récupérée
- l'identifiant de verrouillage de Dossier est initialisé dans le cas de demande de verrouillage

Si cette action est non valide :

- ajout de l'erreur dans l'objet erreur
- émission d'un événement d'erreur en fonction du type de cette erreur
- l'identifiant de verrouillage de Dossier est initialisé avec une valeur vide dans le cas de demande de verrouillage et le Dossier passe en état 'non-modifiable'

Dans tous les cas :

- les compteurs de temps de conversation sont initialisés
- les attributs d'informations contextuelles – s'ils sont présents – sont initialisés

Cette action est systématiquement disponible sur un nœud racine.

Java

- Signature
```
public void readInstanceWithAllChildren() throws
LocalException, ServerException, CommunicationError,
SystemError
public void readInstanceWithAllChildrenAndLock() throws
LocalException, ServerException, CommunicationError,
SystemError
```
- Nom d'utilisation
```
Read Instance With All Children
Read Instance With All Children And Lock
```

Smalltalk

- Signature
```
readInstanceWithAllChildren[AndLock]
```
- Nom d'utilisation
```
readInstanceWithAllChildren[AndLock]
```

COM

- Signature C++
```
public void readWithAllChildren()
public void readWithAllChildrenAndLock()
```
- Nom d'utilisation
```
readWithAllChildren
readWithAllChildrenAndLock
```

3.2.2.3. Lecture de la hiérarchie immédiate d'une instance courante

Comportement

Cette action permet de récupérer tout ou partie des instances des nœuds dépendants de premier niveau hiérarchique du nœud sur lequel est exécutée l'action, en fonction de l'instance présente dans son attribut **présentation d'une instance**.

Cette action est valide si :

- l'attribut présentation d'une instance du nœud contient une instance

Si cette action est valide, son résultat est identique à celui de l'action de lecture d'une instance et de sa hiérarchie immédiate.

Cette action est systématiquement disponible sur tous les nœuds racine et dépendants.

Java

- Signature
```
public void readFirstChildrenFromCurrentInstance() throws
LocalException, ServerException, CommunicationError,
SystemError
```
- Nom d'utilisation
```
Read First Children From Detail
```

Smalltalk

- Signature
```
readFirstChildrenFromCurrentInstance
```
- Nom d'utilisation
```
readFirstChildrenFromCurrentInstance
```

COM

- Signature C++
```
public void readFirstChildrenFromDetail()
```
- Nom d'utilisation
```
readFirstChildrenFromDetail
```

3.2.2.4. Lecture de la hiérarchie complète d'une instance courante

Comportement

Cette action permet de récupérer toutes les instances des nœuds dépendants de tout le Dossier en fonction de l'instance présente dans l'atttribut **présentation d'une instance** d'un nœud racine.

Cette action est valide si :

- l'attribut présentation d'une instance du nœud racine contient une instance

Si cette action est valide son résultat est identique à celui de l'action de lecture d'une instance et de sa hiérarchie complète.

Cette action est systématiquement disponible sur tous les nœuds racine.

Java

- Signature `public void readAllChildrenFromCurrentInstance() throws LocalException, ServerException, CommunicationError, SystemError`
- Nom d'utilisation `Read All Children From Detail`

Smalltalk

- Signature `readAllChildrenFromCurrentInstance`
- Nom d'utilisation `readAllChildrenFromCurrentInstance`

COM

- Signature C++ `public void readAllChildrenFromDetail()`
- Nom d'utilisation `ReadAllChildrenFromDetail`

3.2.2.5. Lecture de la hiérarchie immédiate d'une instance par anticipation

Comportement

Cette action a les mêmes fonctionnalités que la lecture d'une instance et de sa hiérarchie immédiate mais permet sans influencer l'interface graphique de sélectionner des instances par anticipation.

Cette action est valide si :

- l'instance passée en paramètre n'a pas une valeur vide

Si cette action est valide :

- les règles sont les mêmes que pour l'action lecture d'une instance et de sa hiérarchie immédiate sauf que l'attribut de présentation d'instance du nœud concerné peut contenir une valeur vide

Si cette action est non valide :

- ajout de l'erreur dans l'objet erreur
- émission d'un événement d'erreur en fonction du type de cette erreur

Cette action est systématiquement disponible sur un nœud racine et dépendant possédant au moins un nœud dépendant.

Java

- Signature
```
public void readFirstChildren({Nom de classe générée}Data d) throws LocalException, ServerException, CommunicationError, SystemError
```
- Nom d'utilisation `Read First Children From {Nom de classe générée}Data`

Smalltalk

- Signature `readFirstChildrenFrom: aDataDescription`
- Nom d'utilisation `readFirstChildrenFrom:`

COM

- Signature C++
```
public void readWithFirstChildrenFrom({Nom de classe générée}Data d)
```
- Nom d'utilisation `readWithFirstChildrenFrom({Nom de classe générée}Data d)`

3.2.2.6. Lecture de la hiérarchie complète d'une instance par anticipation

Comportement

Cette action a les mêmes fonctionnalités que la lecture d'une instance et de sa hiérarchie complète mais permet sans influencer l'interface graphique de sélectionner des instances par anticipation.

Cette action est valide si :

- l'instance passée en paramètre n'a pas une valeur vide

Si cette action est valide :

- les règles sont les mêmes que pour l'action lecture d'une instance et de sa hiérarchie complète sauf que l'attribut de présentation d'instance du nœud concerné peut contenir une valeur vide

Si cette action est non valide :

- ajout de l'erreur dans l'objet erreur
- émission d'un événement d'erreur en fonction du type de cette erreur

Cette action est systématiquement disponible sur un nœud racine.

Java

- Signature `public void readAllChildren({Nom de classe générée}Data d) throws LocalException, ServerException, CommunicationError, SystemError`
- Nom d'utilisation `Read All Children From {Nom de classe générée}Data`

Smalltalk

- Signature `readAllChildrenFrom: aDataDescription`
- Nom d'utilisation `readAllChildrenFrom:`

COM

- Signature C++ `public void readWithAllChildrenFrom({Nom de classe générée}Data d)`
- Nom d'utilisation `readWithAllChildrenFrom({Nom de classe générée}Data d)`

3.2.3. Gestion de la pagination

3.2.3.1. Lecture des instances de la page suivante

Comportement

Cette action permet de récupérer la page suivante d'une collection d'un nœud. Lorsque la pagination choisie est de type `extend`, les instances récupérées sont cumulées aux instances déjà existantes dans l'attribut **présentation d'une liste d'instances**. Les instances créées localement qui pourraient être en conflit avec les instances récupérées sont prioritaires. Lorsque la pagination est de type `non-extend`, les instances contenues dans l'attribut **présentation d'une liste d'instances** sont écrasées par les instances récupérées.

Cette action est valide si :

- la dernière page de la collection n'a pas déjà été atteinte. Dans le cas contraire, cette action ne provoque pas d'accès au serveur et émet l'événement de récupération de la dernière page de collection.
- sur un nœud dépendant, une collection doit avoir été définie précédemment ou l'attribut **présentation d'une instance** du nœud père doit contenir une instance.
- sur un nœud racine ou référence, si une collection n'a pas été définie, cette action se comporte comme une action de sélection d'instances.

Si cette action est valide :

- l'attribut présentation d'une liste d'instance est initialisé par le résultat de la requête en fonction du type de pagination et du mode de gestion de la collection
- si la pagination concerne un nœud racine et est de type `non-extend`, avec un mode de gestion de collection automatique, l'attribut de présentation d'instance de ce nœud est initialisée avec une valeur vide
- si la pagination concerne un nœud racine ou dépendant et est de type `extend`, ou avec un mode de gestion de collection automatique, son attribut de présentation d'instance ainsi que les attributs de présentation d'instance et de listes d'instances des nœuds dépendants ne sont pas modifiés
- le compteur du nombre total d'instances locales est initialisé
- le libellé et la clé du message de retour de sélection du nœud sont initialisés si la dernière instance de la collection a été récupérée

- émission d'un événement de non détection d'erreur
- émission de l'événement présence d'instances locales en mise à jour, en mode de gestion de collection automatique, si des instances en mises à jour sont toujours présentes dans le cache local
- émission de l'événement de récupération de la première page d'une collection si l'action concerne le nœud racine ou référence, si le type de pagination est **non-extend**, avec un mode de gestion de collection automatique et si c'est la première page récupérée de la collection
- émission de l'événement de présence d'au moins une page précédente si l'action concerne le nœud racine ou référence, si le type de pagination est **non-extend**, avec un mode de gestion de collection automatique et si ce n'est pas la première page récupérée de la collection
- émission de l'événement de présence d'au moins une page suivante si la dernière instance de la collection n'a pas été récupérée
- émission de l'événement de récupération de la dernière page si la dernière instance de la collection a été récupérée

Si cette action est non valide :

- ajout de l'erreur dans l'objet erreur
- émission d'un événement d'erreur en fonction du type de cette erreur

Dans tous les cas :

- les compteurs de temps de conversation sont initialisés
- les attributs d'informations contextuelles – s'ils sont présents – sont initialisés

Cette action est systématiquement disponible sur tous les nœuds.

Java

- Signature `public void readNextPage() throws LocalException, ServerException, CommunicationError, SystemError`
- Nom d'utilisation `Read Next Page`

Smalltalk

- Signature `readNextPage`
- Nom d'utilisation `readNextPage`

COM

- Signature C++ `public void readNextPage()`
- Nom d'utilisation `readNextPage`

3.2.3.2. Lecture des instances de la page précédente

Comportement

Cette action permet de récupérer la page précédente d'une collection d'un nœud. Cette action est réservée exclusivement à la pagination de type **non-extend** avec un mode de gestion de collection automatique. Les instances présentes dans l'attribut **présentation d'une liste d'instances** sont systématiquement écrasées par les instances récupérées.

Cette action est valide si :

- la première page de la collection n'a pas déjà été atteinte. Dans le cas contraire, cette action ne provoque pas d'accès au serveur et émet l'événement de récupération de la première page de collection.

- si une collection n’a pas été définie, cette action se comporte comme une action de sélection d’instances.

Si cette action est valide :

- l’attribut de présentation d’une liste d’instances est initialisé
- l’attribut de présentation d’une liste d’instances est modifié
- le compteur du nombre total d’instances locales est initialisé
- le libellé et la clé du message de retour de sélection sont initialisés si la dernière instance de la collection a été récupérée
- les attributs de présentation d’une instance et de liste d’instance des nœuds dépendants sont initialisés avec une valeur vide
- émission d’un événement de non détection d’erreur
- émission de l’événement présence d’instances locales en mise à jour si des instances en mises à jour sont toujours présentes dans le cache local
- émission de l’événement de présence d’au moins une page précédente si ce n’est pas la première page récupérée de la collection
- émission de l’événement de récupération de la première page d’une collection si c’est la première page de la collection
- émission de l’événement de présence d’au moins une page suivante si la dernière instance de la collection n’a pas été récupérée
- émission de l’événement de récupération de la dernière page si la dernière instance de la collection a été récupérée

Si cette action est non valide :

- ajout de l’erreur dans l’objet erreur
- émission d’un événement d’erreur en fonction du type de cette erreur

Dans tous les cas :

- les compteurs de temps de conversation sont initialisés
- les attributs d’informations contextuelles – s'ils sont présents – sont initialisés

Cette action est disponible sur les nœuds racines et références.

Java

- Signature `public void readPreviousPage() throws LocalException, ServerException, CommunicationError, SystemError`
- Nom d’utilisation `Read Previous Page`

Smalltalk

- Signature `readPreviousPage`
- Nom d’utilisation `readPreviousPage`

COM

- Signature C++ `public void readPreviousPage()`
- Nom d’utilisation `readPreviousPage`

3.2.4. Emission des mises à jour

3.2.4.1. Emission des mises à jour locales sur le serveur

Comportement

Cette action permet d'envoyer au serveur toutes les mises à jour effectuées localement depuis la dernière exécution de la même action.

Seuls les mouvements utiles sont envoyés.

Cette action est valide si :

- au moins une mise à jour locale a été effectuée.

Si cette action est valide :

- toutes les instances mises à jour sont supprimées du cache local
- chaque instance de Vue Logique modifiée est réactualisée dans le cache local par sa dernière image serveur après mise à jour si l'option de rafraîchissement des instances est positionnée à l'émission de l'action
- le contrôle des données sur le serveur peut être activé en positionnant l'attribut appropriée avant l'exécution de l'action.
- émission d'un événement de non détection d'erreur

Si cette action est non valide :

- ajout de l'erreur dans l'objet erreur
- émission d'un événement d'erreur en fonction du type de cette erreur

Dans tous les cas :

- les compteurs de temps de conversation sont initialisés
- les attributs d'informations contextuelles – s'ils sont présents – sont initialisés

Cette action est disponible sur un nœud racine lorsqu'au moins un des Composants Applicatifs associés aux nœuds du Dossier peut effectuer des mises à jour.

Java

- Signature
```
public void updateFolder() throws LocalException,
ServerException, CommunicationError, SystemError
```
- Nom d'utilisation `Update Folder`

Smalltalk

- Signature `updateFolder`
- Nom d'utilisation `updateFolder`

COM

- Signature C++ `public void updateFolder()`
- Nom d'utilisation `updateFolder`

3.2.4.2. Emission des mises à jour et sélection

Comportement

Cette action permet d'envoyer au serveur toutes les mises à jour effectuées localement depuis la dernière exécution de la même action et demande une nouvelle sélection de collection d'instances du nœud racine à la suite du traitement de la mise à jour si celui-ci ne détecte pas d'erreur.

Seuls les mouvements utiles sont envoyés.

Cette action est valide si :

- au moins une mise à jour locale a été effectuée
- le rafraîchissement des données sur le serveur n'est pas activé

Si cette action est valide :

- toutes les instances mises à jour sont supprimées du cache local
- émission d'un événement de non détection d'erreur
- toutes les règles définies dans l'action de sélection d'un ensemble d'instances sont activées

Si cette action est non valide :

- ajout de l'erreur dans l'objet erreur
- émission d'un événement d'erreur en fonction du type de cette erreur

Dans tous les cas :

- les compteurs de temps de conversation sont initialisés
- les attributs d'informations contextuelles – s'ils sont présents – sont initialisés

Cette action est disponible sur un nœud racine lorsqu'au moins un des Composants Applicatifs associés aux nœuds du Dossier peut effectuer des mises à jour.

Java

Non disponible.

Smalltalk

- Signature `updateFolderAndSelectInstances`
- Nom d'utilisation `updateFolderAndSelectInstances`

COM

Non disponible.

3.2.5. Gestion du verrouillage logique

3.2.5.1. Verrouillage logique d'une instance courante

Comportement

Cette action permet de s'approprier une instance de Dossier en mise à jour exclusive. Cette action peut porter sur une instance locale qui n'existe pas dans la base de données.

Cette action est valide si :

- l'attribut Critères de sélection du nœud racine contient l'identifiant d'une instance de Vue Logique
- l'instance n'est pas déjà verrouillée

Si cette action est valide :

- émission d'un événement de non détection d'erreur
- l'attribut identifiant de verrouillage du Dossier est initialisé avec la valeur retournée par le serveur
- le Dossier passe en état « modifiable »

Si cette action est non valide :

- ajout de l'erreur dans l'objet erreur
- émission d'un événement d'erreur en fonction du type de cette erreur
- l'attribut identifiant de verrouillage du Dossier est initialisé avec une valeur vide
- le Dossier passe en état « non modifiable »

Dans tous les cas :

- les compteurs de temps de conversation sont initialisés
- les attributs d'informations contextuelles – s'ils sont présents – sont initialisés

Cette action est disponible sur un nœud racine lorsque l'option verrouillage logique est positionnée dans le Référentiel VisualAge Pacbase pour le Dossier concerné.

Java

- Signature
```
public void lock() throws LocalException,
ServerException, SystemError, CommunicationError
```
- Nom d'utilisation `Lock`

Smalltalk

- Signature `lock`
- Nom d'utilisation `lock`

COM

- Signature C++ `public void lock()`
- Nom d'utilisation `lock`

3.2.5.2. Déverrouillage logique d'une instance courante

Comportement

Cette action permet de libérer une instance de Dossier utilisée en mise à jour exclusive lorsque l'utilisateur décide de ne pas envoyer les instances mises à jour en local vers le serveur.

Cette action est valide si :

- l'attribut Critères de sélection du nœud racine contient l'identifiant d'une instance de Vue Logique
- l'instance est verrouillée

Si cette action est valide :

- émission d'un événement de non détection d'erreur
- l'attribut identifiant de verrouillage du Dossier est initialisé avec la valeur vide

Si cette action est non valide :

- ajout de l'erreur dans l'objet erreur
- émission d'un événement d'erreur en fonction du type de cette erreur

Dans tous les cas :

- les compteurs de temps de conversation sont initialisés
- les attributs d'informations contextuelles – s'ils sont présents – sont initialisés

Cette action est disponible sur un nœud racine lorsque l'option verrouillage logique est positionnée dans le Référentiel VisualAge Pacbase pour le Dossier concerné.

Java

- Signature
```
public void unlock() throws LocalException,
ServerException, SystemError, CommunicationError
```
- Nom d'utilisation `Unlock`

Smalltalk

- Signature `unlock`
- Nom d'utilisation `unlock`

COM

- Signature C++ `public void unlock()`
- Nom d'utilisation `unlock`

3.2.6. Actions de connexion et déconnexion au serveur

3.2.6.1. Connexion au serveur

Comportement

Cette action permet à l'utilisateur d'effectuer une connexion explicite au serveur lui permettant, entre autres, de vérifier la disponibilité de la communication avant toute émission de requête fonctionnelle.

Java

Non disponible.

Smalltalk

- Signature `connect`
- Nom d'utilisation `connect`

COM

Non disponible.

3.2.6.2. Déconnexion au serveur

Comportement

Cette action permet de libérer explicitement les ressources utilisées par le middleware avant de sortir d'une application.

Java

Non disponible.

Smalltalk

- Signature `disconnect`
- Nom d'utilisation `disconnect`

COM

Non disponible.

3.2.7. Gestion des instances dépendantes

3.2.7.1. Contrôle de présence d'instances dépendantes

Comportement

Cette action permet de savoir si l'instance de Vue Logique contenue dans l'attribut **présentation d'une instance** du nœud possède des instances dépendantes. Si cette instance n'a pas été créée localement et ne contient pas en local d'instances dépendantes, le système émet cette action sur le serveur pour vérifier l'existence d'instances dépendantes de premier niveau hiérarchique.

Cette action est valide si :

- l'attribut présentation d'une instance contient une valeur non vide

Si cette action est valide :

- émission d'un événement de non détection d'erreur
- émission d'un événement présence d'une instance dépendante ou d'un événement absence d'une instance dépendante

Si cette action est non valide :

- ajout de l'erreur dans l'objet erreur
- émission d'un événement d'erreur en fonction du type de cette erreur

Dans tous les cas si l'action a été transmise au serveur :

- les compteurs de temps de conversation sont initialisés
- les attributs d'informations contextuelles – s'ils sont présents – sont initialisés

Cette action est disponible lorsque le nœud concerné a au moins un nœud dépendant.

Java

- Signature `public void checkExistenceOfDependentInstances() throws LocalException, ServerException, CommunicationError, SystemError`
- Nom d'utilisation `Check existence of dependent instances`

Smalltalk

- Signature `checkExistenceOfDependentInstances`
- Nom d'utilisation `checkExistenceOfDependentInstances`

COM

- Signature C++ `public void checkExistenceOfDependencies()`
- Nom d'utilisation `checkExistenceOfDependencies`

3.2.8. Gestion des Services Utilisateur

3.2.8.1. Exécution de Services Utilisateur

Comportement

Cette action permet d'exécuter un Service Utilisateur associé au nœud ainsi qu'à tous ses nœuds dépendants qui ont positionné un Service Utilisateur à exécuter.

Cette action est effective si :

- au moins un des nœuds concernés contient une valeur non vide dans l'attribut Service Utilisateur à exécuter

Si cette action est valide :

- émission d'un événement de non détection d'erreur
- l'attribut présentation d'instances rendues par un Service Utilisateur est initialisé
- l'attribut nombre d'instances de Vue Logique traitées par un Service Utilisateur est recalculé

Si cette action est non valide :

- ajout de l'erreur dans l'objet erreur
- émission d'un événement d'erreur en fonction du type de cette erreur

Dans tous les cas :

- les compteurs de temps de conversation sont initialisés
- les attributs d'informations contextuelles – s'ils sont présents – sont initialisés

Cette action est disponible si le Composant Applicatif associé au nœud contient au moins un Service Utilisateur.

Java

- Signature `public void executeUserService() throws ServerException, CommunicationError, SystemError, LocalException`
- Nom d'utilisation `Execute User Service`

Smalltalk

- Signature `executeUserService`
- Nom d'utilisation `executeUserService`

COM

- Signature C++ `public void executeUserService()`
- Nom d'utilisation `executeUserService`

3.2.9. Gestion des conversations asynchrones

3.2.9.1. Récupération différée d'une réponse

Comportement

Cette action permet de récupérer la réponse associée à une requête précédemment émise avec un type de communication asynchrone.

Cette action est valide si :

- le protocole de communication utilisé pour émettre la requête permet les conversations asynchrones
- le type de conversation est asynchrone
- l'identifiant de la requête passée en paramètre est valide et connu

Si cette action est valide et la requête disponible :

- les règles utilisées sont les mêmes que celles définies pour l'action ayant émis la requête lorsque celle-ci est exécutée en mode synchrone
- l'attribut de nombre de réponses en attente est décrémenté de 1
- les attributs d'informations contextuelles – s'ils sont présents – sont initialisés

Si cette action est non valide :

- ajout de l'erreur dans l'objet erreur
- émission d'un événement d'erreur en fonction du type de cette erreur

Si la requête n'est pas disponible :

- émission de l'événement de récupération de réponse non disponible

Dans tous les cas :

- les compteurs de temps de conversation sont initialisés

Cette action est systématiquement disponible sur un nœud racine.

Java

- Signature `public Boolean getReply(com.ibm.vap.generic.ServerActionContexts) throws LocalException, ServerException, CommunicationError, SystemError`
- Nom d'utilisation `Get Reply`

Smalltalk

- Signature `getReplyOf: aReplyId`
- Nom d'utilisation `getReplyOf:`

COM

- Signature C++ `public Boolean getReply(ServerActionContext s)`
- Nom d'utilisation `getReply(ServerActionContext s)`

3.2.9.2. Elimination d'une réponse en attente de récupération

Comportement

Cette action permet de supprimer explicitement un identifiant de réponse associé à une requête.

Cette action est valide si :

- le protocole de communication utilisé pour émettre la requête permet les conversations asynchrones
- le type de conversation est asynchrone
- l'identifiant de la requête passée en paramètre est valide et connu

Si cette action est valide et la requête disponible :

- l'identifiant de réponse est supprimé et la réponse correspondante est purgée du gestionnaire de message
- l'attribut de nombre de réponses en attente est décrémenté de 1

Si cette action est non valide :

- ajout de l'erreur dans l'objet erreur
- émission d'un événement d'erreur en fonction du type de cette erreur

Si la requête n'est pas disponible :

- émission de l'événement de récupération de réponse non disponible

Dans tous les cas :

- les compteurs de temps de conversation sont initialisés

Cette action est systématiquement disponible sur un nœud racine.

Java

Non disponible.

Smalltalk

- Signature `resetPendingReplyOf: aReplyId`
- Nom d'utilisation `resetPendingReplyOf:`

COM

Non disponible.

3.2.9.3. Contrôle de la validité d'un identifiant de message

Comportement

Cette action permet de savoir si un identifiant de requête est valide et connu.

Si cette action est valide :

- cette action revoie `true` si l'identifiant est valide, `false` sinon.

Si cette action est non valide :

- ajout de l'erreur dans l'objet erreur
- émission d'un événement d'erreur en fonction du type de cette erreur

Si la requête n'est pas disponible :

- émission de l'événement de récupération de réponse non disponible

Cette action est systématiquement disponible sur un nœud racine.

Java

- Signature `public Boolean isReplyValid(com.ibm.vap.generic.ServerActionContext aContext)`
- Nom d'utilisation `Checks the validity of the request reply`

Smalltalk

- Signature `isReplyIdValid: aReplyId`
- Nom d'utilisation `isReplyIdValid:`

COM

- Signature C++ `public Boolean isReplyValid(ServerActionContext s)`
- Nom d'utilisation `isReplyValid(ServerActionContext s)`

3.2.10. Gestion des actions exécutées

3.2.10.1. Réexécution de la dernière action

Comportement

Cette action permet de ré-exécuter la dernière action exécutée localement ou sur le serveur.

Les règles de validité et de traitements au retour de cette action sont celles de l'action qui s'exécute de nouveau.

Java

Non disponible.

Smalltalk

- Signature `redoLastAction`
- Nom d'utilisation `redoLastAction`

COM

Non disponible.

3.2.11. Gestion des sous-schémas

Comportement

Cette action permet de récupérer, par appel du Composant Applicatif correspondant à la Vue Logique, les valeurs des Rubriques n'appartenant pas au sous-schéma sélectionné avec l'attribut **`subSchema`**.

Au retour correct de cette action, l'instance est considérée comme complète et donc son sous-schéma implicite associé est réinitialisé. Toute modification ultérieure s'effectue donc sans sous-schéma associé.

Avant cette action, les Rubriques appartenant au sous-schéma ont pu être modifiées localement.

Cette action est disponible si les Composants Applicatifs gèrent la présence des Rubriques (**`VECTPRES=YES`** ou **`CHECKSER=YES`**) et si le nœud comporte au moins un sous-schéma.

Java

- Signature
```
public completeInstance throws LocalException,
ServerException, CommunicationError, SystemError
```
- Nom d'utilisation `Complete Instance`

Smalltalk

- Signature `completeInstance`
- Nom d'utilisation `completeInstance`

COM

- Signature C++ `public completeInstance()`
- Nom d'utilisation `completeInstance`

4. Evénements

4.1. Gestion de la pagination

Pour la cible COM, tous les événements décrits dans ce chapitre sont à récupérer dans la pile d'événements associée à chaque Proxy par l'intermédiaire de la méthode `public String popServerEvent()` tant que `public Int getServerEventsCount()` ne renvoie pas zéro. Les String récupérés correspondent aux codes des événements décrits.

4.1.1. Signal de récupération de la dernière page d'une collection

Règles d'émission

Ce signal est émis par un nœud lorsqu'une action de sélection d'un ensemble d'instances ou de pagination renvoie une page qui contient la dernière instance de la collection sélectionnée. Cet événement est disponible pour les nœuds racine, références et pour les nœuds dépendants de cardinalité maximale 1.

Java

- Code `noPageAfter`

Smalltalk

- Code `noPageAfter`

COM

- Code du String `NO_PAGE_AFTER`

4.1.2. Signal de récupération de la première page d'une collection

Règles d'émission

Ce signal est émis par un nœud lorsqu'une action de sélection d'un ensemble d'instances ou de pagination renvoie une page qui contient la première instance de la collection sélectionnée. Cet événement est disponible pour les nœuds racine et références lorsque le mode de pagination est de type `non-extend` avec un mode de gestion de collection automatique.

Java

- Code `noPageBefore`

Smalltalk

- Code `noPageBefore`

COM

- Code du String `NO_PAGE_BEFORE`

4.1.3. Signal de présence d'au moins une page suivante

Règles d'émission

Ce signal est émis par un nœud lorsqu'une action de sélection d'un ensemble d'instances ou de pagination renvoie une page qui ne contient pas la dernière instance de la collection sélectionnée. Cet événement est disponible pour les nœuds racine, références et pour les nœuds dépendants de cardinalité maximale 1.

Java

- Code `pageAfter`

Smalltalk

- Code `pageAfter`

COM

- Code du String `PAGE_AFTER`

4.1.4. Signal de présence d'au moins une page précédente

Règles d'émission

Ce signal est émis par un nœud lorsqu'une action de sélection d'un ensemble d'instances ou de pagination renvoie une page qui ne contient pas la première instance de la collection sélectionnée.
Cet événement est disponible pour les nœuds racine et références lorsque le mode de pagination est de type `non-extend` avec un mode de gestion de collection automatique.

Java

- Code `pageBefore`

Smalltalk

- Code `pageBefore`

COM

- Code du String `PAGE_BEFORE`

4.2. Gestion des lectures unitaires

4.2.1. Signal de lecture d'un enregistrement non trouvé

Règles d'émission

Ce signal est émis par un nœud lorsqu'une action de lecture d'une instance ne renvoie pas l'instance demandée.

Cette action est systématiquement disponible pour tous les nœuds.

Java

- Code `notFound`

Smalltalk

- Code `notFound`

COM

- Code du String `NOT_FOUND`

4.3. Gestion des sélections simultanées

4.3.1. Signal de non participation à une lecture simultanée

Règles d'émission

Ce signal est émis par un nœud suite à l'action de récupération du dernier événement associé à la dernière sélection serveur, lorsque ce nœud n'a pas participé à cette action de sélection.

Cette action est systématiquement disponible pour tous les nœuds.

Java

- Code `notRead`

Smalltalk

- Code `notRead`

COM

- Code du String `NOT_READ`

4.4. Gestion du verrouillage logique

4.4.1. Signal de verrouillage logique effectué

Règles d'émission

Ce signal est émis par un nœud racine après une action de demande de verrouillage logique d'une instance qui est valide.

Cette action est disponible pour un nœud racine lorsque l'option de verrouillage logique a été codifiée dans le Référentiel VisualAge Pacbase pour le nœud concerné.

Java

- Code `lockSuccessful`

Smalltalk

- Code `lockSuccessful`

COM

- Code du String `LOCK_SUCCESSFUL`

4.4.2. Signal de verrouillage logique infructueux

Règles d'émission

Ce signal est émis par un nœud racine après une action de demande de verrouillage logique d'une instance lorsque cette instance est déjà en mise à jour exclusive par un autre utilisateur.

Cette action est disponible pour un nœud racine lorsque l'option de verrouillage logique a été codifiée dans le Référentiel VisualAge Pacbase pour le nœud concerné.

Java

- Code `lockFailed`

Smalltalk

- Code `lockFailed`

COM

- Code du String `LOCK_FAILED`

4.5. Gestion des instances dépendantes

4.5.1. Signal de présence d'au moins une instance dépendante

Règles d'émission

Ce signal est émis par un nœud après une action de contrôle de présence d'instances dépendantes lorsque l'instance concernée possède au moins une instance dépendante. Cette action est systématiquement disponible pour les nœuds racine et dépendants.

Java

- Code `dependentInstances`

Smalltalk

- Code `dependentInstances`

COM

- Code du String `DEPENDENT_INSTANCES`

4.5.2. Signal d'absence d'instance dépendante

Règles d'émission

Ce signal est émis par un nœud après une action de contrôle de présence d'instances dépendantes lorsque l'instance concernée ne possède aucune instance dépendante. Cette action est systématiquement disponible pour les nœuds racine et dépendants.

Java

- Code `noDependentInstances`

Smalltalk

- Code `noDependentInstances`

COM

- Code du String `NO_DEPENDENT_INSTANCES`

4.6. Gestion des conversations asynchrones

4.6.1. Signal d'exécution d'une action serveur asynchrone

Règles d'émission

Ce signal est émis par un nœud racine lorsque la requête traitée par le serveur utilise un type de conversation asynchrone. Cette action est systématiquement disponible pour un nœud racine.

Java

Remplacé par exception `AsynchronousRequestException`

Smalltalk

- Code `asyncRequest`

COM

Non disponible.

4.6.2. Signal de récupération d'une réponse non disponible

Règles d'émission

Ce signal est émis par un nœud racine lorsque la réponse associée à l'identifiant de réponse de la requête n'est pas disponible au niveau du gestionnaire de messages.

Cette action est systématiquement disponible pour un nœud racine.

Java

Remplacé par la valeur `false` retournée par la méthode `getReply`

Smalltalk

- Code `replyPending`

COM

Remplacé par la valeur `false` retournée par la méthode `getReply`

4.7. Gestion d’une collection d’instances de Vue Logique

4.7.1. Signal avant changement d’une collection d’instances

Règles d’émission

Ce signal est émis lorsqu’au moins une mise à jour locale a été effectuée sur la collection d’instance en cours et qu’une nouvelle action de sélection d’instances ou de pagination est faite.

Il permet de prévenir un éventuel oubli de mise à jour sur le serveur.

Java

- Code `aboutToChangeSelection`

Smalltalk

- Code `aboutToChangeSelection`

COM

- Code du String `ABOUT_TO_CHANGE_SELECTION`

4.8. Gestion des erreurs

4.8.1. Signal de restauration du contexte lié à l’erreur

Règles d’émission

Ce signal est émis par un nœud lorsque, suite à une demande de restauration de contexte associé à une erreur, cette demande s’est effectuée correctement, c’est à dire que l’attribut **présentation d’une instance** ou **présentation d’une instance liée à un Service Utilisateur** a été initialisé par l’instance ayant provoqué l’erreur.

Cette action est systématiquement disponible pour un nœud racine ou dépendant.

Java

Non disponible.

Smalltalk

- Code `errorContextRestored`

COM

Non disponible.

5. Interface publique de manipulation des Rubriques

5.1. Gestion du contenu de la Rubrique

Description

Cet attribut donne le contenu de la Rubrique.

Cet attribut est systématiquement disponible pour les Rubriques définies dans les classes **`DataDescription`**, **`SelectionCriteria`** et **`UserContext`**.

Java

- Type — Dépend du type de la Rubrique (**`java.lang.String`**, **`int`**, **`long`**, **`double`**, ou **`java.util.Date`**)
- Code interne — **`<CodeRubrique>`**
- Nom d'utilisation — **`<Nom en clair de la Rubrique>`**
- get/set —
```
public [type] get<CodeRubrique> ()
public void set<CodeRubrique> (Type t )
```

Smalltalk

En plus de la gestion implicite via le *QuickForm*, cet attribut peut être géré explicitement :

- Type — **`VpcsString`**, **`VpcsInteger`**, **`VpcsDecimal`**, **`VpcsDate`**, ou **`VpcsTime`**
- Code interne — **`<CodeRubrique>`**
- Nom d'utilisation — **`<CodeRubrique>`**
- get/set — **`<CodeRubrique>`** / **`<CodeRubrique>:`**

COM

- Type — Dépend du type de la Rubrique
- Code interne — **`<CodeRubrique>`**
- Nom d'utilisation — **`<CodeRubrique>`**
- get/set C++ —
```
public [Type] get<CodeRubrique> ()
public void set<CodeRubrique> (Type t )
```

5.2. Gestion des codes des valeurs permises

Description

Cet attribut donne les valeurs permises associées à une Rubrique.

Cet attribut est systématiquement disponible pour les Rubriques qui contiennent des valeurs permises et qui sont définies dans la classe **`DataDescription`**.

☞ Pour l'environnement VisualAge Smalltalk, les Rubriques sont manipulées implicitement via le *QuickForm*.

Java

- Type `[ Type [ ] ]`
- Code interne `<CodeRubrique>Values`
- Nom d'utilisation `<CodeRubrique>Values`
- get/set `public [Type[ ] ] get<CodeRubrique>Values ()`
  set non disponible

COM

Disponible avec une API de parcours de collection

- Nb d'éléments `public Int <CodeRubrique>ValidValuesCount()`
- Elément `public Type <CodeRubrique>ValidValuesAt(Int i)`

5.3. Gestion des libellés des valeurs permises

Description

Cet attribut donne les libellés des valeurs permises associées à une Rubrique.

Cet attribut est systématiquement disponible pour les Rubriques qui contiennent des valeurs permises et qui sont définies dans la classe **`DataDescription`**.

☞ Pour l'environnement VisualAge Smalltalk, les Rubriques sont manipulées implicitement via le *QuickForm*.

Java

- Type `String [ ]`
- Code interne `<CodeRubrique>Labels`
- Nom d'utilisation `<CodeRubrique> Labels`
- get/set `public String[ ] get<CodeRubrique>Labels ( )`
  set non disponible

COM

Disponible avec une API de parcours de collection

- Nb d'éléments `public Int <CodeRubrique>ValidLabelsCount()`
- Elément `public Type <CodeRubrique>ValidLabelsAt(Int i)`

5.4. Gestion de la validité du contenu d'une Rubrique

Description

Cette action indique si le contenu d'une Rubrique est valide ou non.

Cette action est systématiquement disponible pour les Rubriques qui contiennent des valeurs permises et qui sont définies dans la classe `DataDescription`.

☞ Pour l'environnement VisualAge Smalltalk, les Rubriques sont manipulées implicitement via le *QuickForm*.

Java

- Signature `public Boolean is<CodeRubrique>Valid ()`
- Nom d'utilisation `Is<CodeRubrique> Valid`

Cette méthode est `deprecated` depuis la version 2.5v07, elle est remplacée par l'utilisation de la méthode `get<CodeRubrique>Error`.

- Signature `public DataFieldError get<CodeRubrique>Error ()`
- Nom d'utilisation `<CodeRubrique> Error`

Cette méthode renvoie une instance de `DataFieldError` qui indique la nature de l'erreur détectée sur le champ ou bien la valeur `null` si le contenu de la rubrique est valide.

COM

Non disponible.

5.5. Gestion de la présence d'une Rubrique

Description

Ces actions indiquent si la Rubrique est absente (contenu vide) ou présente (contenu non vide).

La première action est systématiquement disponible pour les Rubriques définies dans les classes **DataDescription** et **UserDataDescription**

- de tous les nœuds pour Java et COM
- des nœuds dont le Composant Applicatif possède l'option **NULLMNGT=YES** pour Smalltalk.

La deuxième action est systématiquement disponible pour les Rubriques définies dans les classes **DataDescription** et **UserDataDescription**

- de tous les nœuds pour Java et COM
- des nœuds dont le Composant Applicatif possède l'option **NULLMNGT=YES** et un service utilisateur ou un service de mise à jour pour Smalltalk.

Avant l'exécution de cette action, toutes les Rubriques sont considérées :

- présentes pour Smalltalk
- absentes pour Java et COM, sauf si une valeur par défaut a été indiquée dans VisualAge Pacbase.

Java

- Signature
```
public Boolean is<CodeRubrique>Present ()
public void set<CodeRubrique>Present (Boolean b)()
```
- Nom d'utilisation
```
<CodeRubrique> Present
```

Smalltalk

- Signature
```
isNull:<CodeRubrique>
setNull: (Boolean b) on: <CodeRubrique>
```
- Nom d'utilisation
```
isNull:
setNull: on:
```

COM

- Signature C++
```
public Boolean is<CodeRubrique>Present ()
public Boolean set<CodeRubrique>Present (Boolean b)
```
- Nom d'utilisation
```
is<CodeRubrique>Present
set<CodeRubrique>Present(Boolean b)
```

5.6. Gestion du contrôle d’une Rubrique

Description

Ces actions indiquent si la Rubrique doit être contrôlée ou non.

Ces actions sont systématiquement disponibles pour les Rubriques définies dans les classes **`DataDescription`** et **`UserDataDescription`** des nœuds racine ou dépendants dont le Composant Applicatif possède les options **`NULLMNGT=YES`** et **`CHECKSER=YES`** et un service de mise à jour.

Avant l'exécution de ces actions, toutes les Rubriques sont considérées comme à contrôler (si l'attribut **`serverCheckOption`** est positionné à true).

Java

- Signature **`public setCheck(int index, boolean aBoolean)`**
- Nom d’utilisation **`Check Flag for the Index’s field`**

L’index de la rubrique à contrôler est retrouvé par l’utilisation de la méthode suivante :

- Signature **`public int get<CodeRubrique>Index()`**
- Nom d’utilisation **`<CodeRubrique> Index`**

Smalltalk

- Signature **`setCheck: aBoolean on: <CodeRubrique>`**
- Nom d’utilisation **`setCheck: on:`**

COM

- Signature C++ **`public setCheck(Int fieldIndex, Boolean b)`**
- Nom d’utilisation get non disponible / **`setCheck(fieldIndex,b)`**

6. Interface publique du gestionnaire d'erreurs [Smalltalk, COM]

Cette interface contient les attributs, actions, événements permettant la gestion de toutes les erreurs locales, de communication et serveur pour les langages Smalltalk et COM.

Pour Smalltalk, le gestionnaire d'erreurs est représenté par l'attribut `errorManager`.

Pour COM, toutes les erreurs sont présentées dans le gestionnaire d'erreurs `VAPERROR`.

6.1. Attributs

6.1.1. Liste d'erreurs

Description

Cet attribut ou action pour COM contient les instances correspondant aux erreurs renvoyées par une action locale ou serveur. Chaque instance contient en particulier les attributs suivants :

- Type de l'erreur
- Action sur laquelle s'est produite l'erreur
- clé complète de l'erreur
- libellé d'erreur
- gravité de l'erreur

Cet attribut est accessible en lecture uniquement.

Smalltalk

- Type `VpcsOrderedCollection` contenant `VpcsError`
- Code interne `errorList`
- Nom d'utilisation `errorList`
- get/set `errorList` / set non disponible

COM

- Signature C++ `public VapError getErrorsElementAt(Int i)`
- Nom d'utilisation `getErrorsElementAt(Int i)`

6.1.2. Gestion du type de l'erreur

Description

Cet attribut pour COM permet de récupérer le type de l'erreur.

- Local
- Server
- Communication

Il est accessible en lecture uniquement.

COM

- Type `String`
- Code interne `getType`
- Nom d'utilisation `getType`
- get/set `public String getType()`/ set non disponible

6.1.3. Gestion de l'action provoquant l'erreur

Description

Cet attribut pour COM permet de récupérer l'action à l'issue de laquelle a été provoquée l'erreur.

Il est accessible en lecture uniquement.

COM

- Type `String`
- Code interne `getAction`
- Nom d'utilisation `getAction`
- get/set `public String getAction ()`/ set non disponible

6.1.4. Gestion de la clef d'erreur

Description

Cet attribut pour COM permet de récupérer la clef de l'erreur (voir le *Guide Utilisateur Volume III – Clients Graphiques* pour la liste des erreurs).

Il est accessible en lecture uniquement.

COM

- Type `String`
- Code interne `getKey`
- Nom d'utilisation `getKey`
- get/set `public String getKey ()`/ set non disponible

6.1.5. Gestion du libellé de l'erreur

Description

Cet attribut pour COM permet de récupérer le libellé de l'erreur.

Il est accessible en lecture uniquement.

COM

- Type `String`
- Code interne `getLabel`
- Nom d'utilisation `getLabel`
- get/set `public String getLabel ()`/ set non disponible

6.1.6. Gestion de la gravité de l'erreur

Description

Cet attribut pour COM permet de récupérer la gravité de l'erreur.

- Exception
- Error

Il est accessible en lecture uniquement.

COM

- Type `String`
- Code interne `getGravity`
- Nom d'utilisation `getGravity`
- get/set `public String getGravity ()`/ set non disponible

6.1.7. Nombre d'erreurs

Description

Cet attribut ou action pour COM contient le nombre d'erreurs renvoyées par une action locale ou serveur.

Il est accessible en lecture uniquement.

Smalltalk

- Type `Integer`
- Code interne `errorCount`
- Nom d'utilisation `errorCount`
- get/set `errorCount` / set non disponible

COM

- Signature C++ `public Int getErrorsCount()`
- Nom d'utilisation `getErrorsCount`

6.2. Actions

6.2.1. Restauration de contexte

Comportement

Cette action permet, à partir d'une clé de message d'erreur, d'alimenter l'attribut de **présentation d'une instance** ou l'attribut de **présentation d'une instance liée à un Service Utilisateur** si le contexte lié à l'erreur peut être restauré. Cette alimentation se fait par l'instance de Vue Logique ayant provoqué l'erreur.

Cette action est valide si :

- la clé complète d'erreur passée en paramètre est valide

Si cette action est valide :

- l'attribut de non récupération du contexte est initialisé à vide ou par le message d'erreur standard si le contexte lié à l'erreur ne peut pas être restauré, et par le message d'erreur standard dans le cas contraire
- l'attribut de présentation d'une instance de Vue Logique ou d'une instance liée à un Service Utilisateur concernée par l'erreur est alimenté si le contexte lié à l'erreur peut être restauré
- les attributs de présentation d'une instance et de liste d'instances des nœuds dépendants sont alimentés suivant la règle globale d'alimentation
- envoi du signal de restauration de contexte lié à l'erreur

Cette action est systématiquement disponible.

Smalltalk

- Signature `restoreContextOfSelectedError: anError`
- Nom d'utilisation `restoreContextOfSelectedError:`

COM

Non disponible.

6.3. Evénements

Pour la cible COM, tous les événements décrits dans ce chapitre sont à récupérer dans la pile d'événements associée à chaque Proxy par l'intermédiaire de la méthode `public String popServerEvent()` tant que `public Int getServerEventsCount()` ne renvoie pas zéro. Les String récupérés correspondent aux codes des événements décrits.

6.3.1. Gestion des erreurs

6.3.1.1. Signal de non détection d'erreur

Règles d'émission

Ce signal est émis par le gestionnaire d'erreurs lorsqu'aucune erreur locale, serveur ou système n'a été détectée.

Cet événement est systématiquement disponible.

Smalltalk

- Code `noError`

COM

- Code `NO_ERROR`

6.3.1.2. Signal de récupération d'une erreur locale

Règles d'émission

Ce signal est émis par le gestionnaire d'erreurs lorsqu'une action locale détecte une erreur.

Cet événement est systématiquement disponible.

Smalltalk

- Code `localError`

COM

- Code `LOCAL_ERROR`

6.3.1.3. Signal de récupération d'une erreur serveur

Règles d'émission

Ce signal est émis par le gestionnaire d'erreurs lorsqu'un serveur détecte une erreur logique d'accès, une erreur utilisateur ou une erreur de contrôle de données de la Vue Logique.

Cet événement est systématiquement disponible.

Smalltalk

- Code `serverError`

COM

- Code `SERVER_ERROR`

6.3.1.4. Signal de récupération d'une erreur système

Règles d'émission

Ce signal est émis par le gestionnaire d'erreurs lorsqu'un serveur détecte une erreur grave comme un déphasage de versions entre le composant client et le Composant Applicatif associé.

Cet événement est systématiquement disponible.

Smalltalk

- Code `systemError`

COM

- Code `SYSTEM_ERROR`

6.3.1.5. Signal de récupération d'une erreur de communication

Règles d'émission

Ce signal est émis par le gestionnaire d'erreurs lorsqu'un problème de communication est détecté au cours d'un échange entre un client et un serveur.

Cet événement est systématiquement disponible.

Smalltalk

- Code `fatalError`

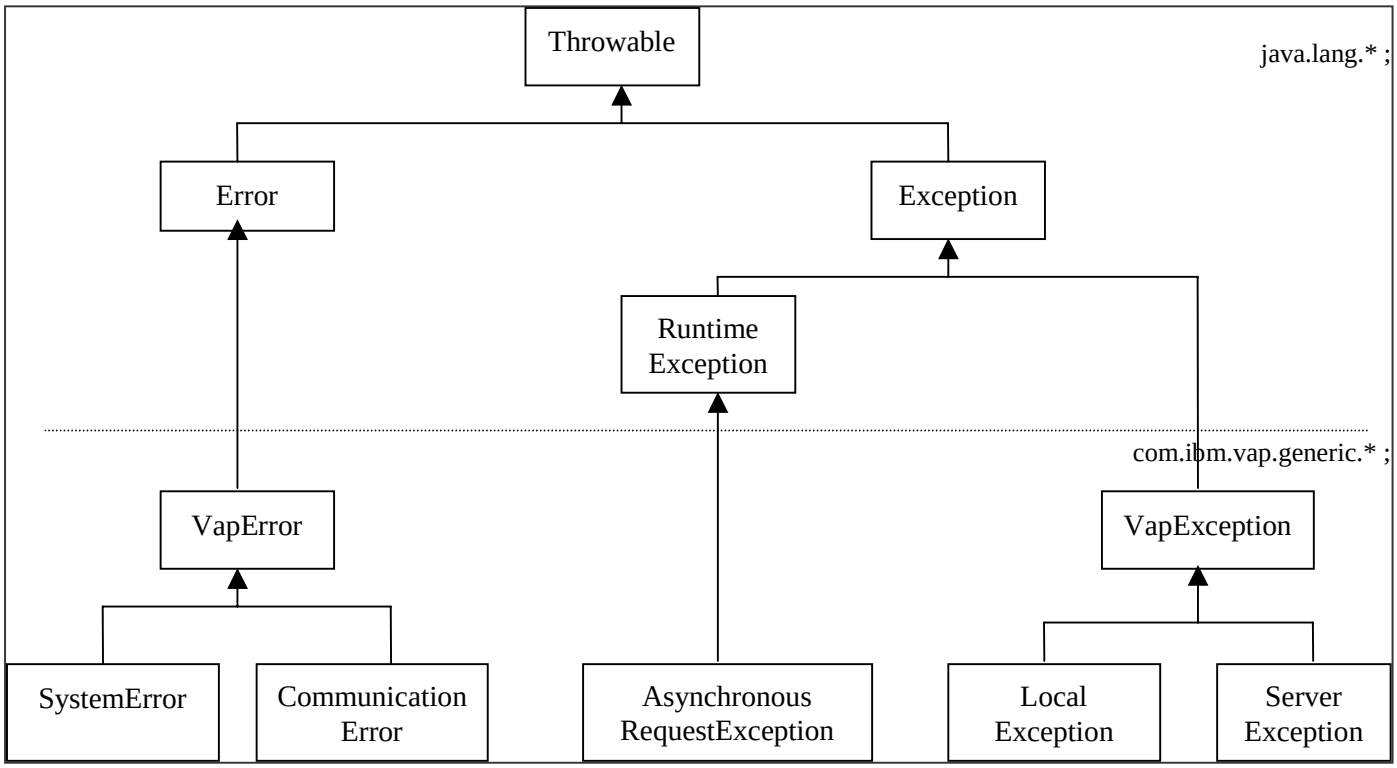
COM

- Code `FATAL_ERROR`

7. Gestion des erreurs pour la cible Java

La gestion de erreurs associées à la manipulation des Proxies Java se base sur le mécanisme des levées d'exceptions, propre à Java.

Quatre classes permettent de véhiculer les erreurs ou exceptions issues d'une Proxy VisualAge Pacbase. Elles sont toutes les quatre à la disposition du développeur ; elles héritent de **`java.lang.Throwable`** de la manière suivante :



L'exception **`com.ibm.vap.generic.AsynchronousRequestException`** signale tout accès serveur, dès lors que la Proxy est en mode asynchrone.

L'exécution de certaines méthodes de la Proxy nécessite de contrôler obligatoirement toute exception héritant de **`com.ibm.vap.generic.VapException`** (les différents types d'exception que peut émettre une méthode sont définies dans la signature documentée dans le manuel).

Il est également fortement recommandé de contrôler les erreurs qui héritent de **`com.ibm.vap.generic.VapError`** bien que le contrôle soit optionnel pour ce type de classe dans Java.

Remarque : L'utilisation abusive de certaines méthodes de la Proxy peut aussi lever l'exception **`java.lang.IllegalStateException`**, en particulier dans le cas de l'utilisation d'une méthode qui est en décalage avec la définition Pacbase de la Proxy (exemple : utilisation de la méthode **`createUserInstance,`** bien qu'aucun service utilisateur n'ait été défini dans la description Pacbase du serveur).

7.1. Classes relatives à la gestion des erreurs

7.1.1. Erreurs de communication

Nom de l'exception

`com.ibm.vap.generic.CommunicationError`

Remarque

Cette exception est provoquée lors d'une erreur dans la chaîne de communication avec le serveur.

Le message porté par chaque instance de cette classe offre des indications quant à l'erreur détectée (méthode `getMessage()` de cette classe).

7.1.2. Erreurs système

Nom de l'exception

`com.ibm.vap.generic.SystemError`

Remarque

Cette exception est provoquée par les erreurs système.

Ce type d'erreur représente une erreur interne et irrécupérable. Elle peut être détectée par le client ou par le serveur.

Dans le cas où le serveur a détecté des erreurs système, les messages associés à ces erreurs sont représentés par des instances de `com.ibm.vap.generic.ServerMessage`. Ces messages sont accessibles par la méthode `java.util.Enumeration serverMessages()` de la classe `com.ibm.vap.generic.SystemError`.

Pour connaître la liste des erreurs système, reportez-vous au manuel *GuideUtilisateur Volume III – Clients Graphiques.*

7.1.3. Erreurs locales

Nom de l'exception

`com.ibm.vap.generic.LocalException`.

Remarque

Cette exception est provoquée par les erreurs locales, détectées par le client.

Cette exception porte une propriété de type int (`int getLocalExceptionKey()`) permettant d'identifier le type de l'erreur qui a conduit à la levée de cette exception (création à tort, instance invalide, …).

Les erreurs à l'origine de cette exception sont décrites le *GuideUtilisateur Volume III – Clients Graphiques* ainsi que dans la documentation HTML associée aux classes génériques : Package `com.ibm.vap.generic`.

7.1.4. Erreurs serveur

Nom de l'exception

`com.ibm.vap.generic.ServerException`.

Remarque

Cette exception est provoquée par les erreurs serveur. Elle est levée à la suite de la réception de message(s) d'erreur logique détecté(s) par le serveur.

Ces messages d'erreurs logiques sont représentés par des instances de `com.ibm.vap.generic.ServerMessage`. La liste de ces message d'erreurs reçus du serveur est accessible *via* la méthode `java.util.Enumeration serverMessages()` de la classe `com.ibm.vap.generic.ServerException`.

Dans le cas d'erreurs détectées lors de services de mise à jour, il est possible de restaurer le contexte de la Proxy relatif à la demande de mise à jour (restauration de l'arbre de sélection et du détail) pour chaque erreur.

La classe `com.ibm.vap.generic.ServerException` indique s'il y a des erreurs « restaurables » dans la liste des erreurs détectées par la méthode (`boolean isContextRestorable()`).

7.1.4.1. Message d'erreur reçu du Serveur

Les messages d'erreur reçus du serveur sont représentés par des objets de type `com.ibm.vap.generic.ServerMessage`.
Cette interface offre des méthodes permettant notament de récupérer la clé d'erreur (`String key()`), le libellé du message d'erreur (`String label()`) et le libellé du message d'erreur formatté par le composant Client (`String localLabel()`).

Pour une description du principe de formattage local des libellés d'erreur, reportez-vous au paragraphe **7.2**, *Personnalisation des libellés des erreurs*, page 126.

7.1.4.2. Message d'erreur reçu du Serveur sur mise à jour

Cette interface hérite de l'interface `com.ibm.vap.generic.ServerMessage`.

Les objets de ce type représentent des messages d'erreur reçus du serveur qui ont été détectés lors de l'exécution de services de mise à jour.

Cette interface offre des méthodes permettant de connaître et de restaurer le contexte de la Proxy relatif à la demande de mise à jour :

- `boolean isContextRestorable`() : indique si le contexte de l'erreur est « restaurable »,
- `void restoreContext() throws LocalException` : provoque la restauration du contexte relatif à l'erreur de mise à jour,
- `DataDescription erroneousData`() : renvoie la classe `DataDescription` dont la mise à jour a échoué,
- `HierarchicalProxyLv erroneousProxy()` : renvoie la classe `HierarchicalProxy` gérant la demande de mise à jour ayant échoué.

7.2. Personnalisation des libellés des erreurs

La personnalisation des libellés des erreurs associées à la manipulation des Proxies est possible avec les Proxies Java. Elle est basée sur l'utilisation de techniques d'internationalisation et de formattage dynamique de libellé fournie par le langage Java : les différents libellés sont stockés dans un fichier de ressources (`vaperror.properties`) qui sera chargé selon le contexte géographique (ce contexte étant fourni par la classe **`Locale`** par défaut). Ce fichier est structuré sur le mode relation clé-valeur, où la valeur correspond à un libellé.

De plus, chacun des libellés doit respecter le formalisme des « *patterns* » utilisé par Java dans le cadre du traitement des libellés à parties variables (**`java.text.MessageFormat`**).

Les libellés d'erreurs stockés dans ce fichier sont :

- le message associé à une instance de **`com.ibm.vap.generic.LocalException`**, **`SystemError`** et **`ServerException`** (accessible par la méthode **`String getMessage()`**),
- le libellé des objets de type **`com.ibm.vap.generic.ServerMessage`** formattés localement (**`String localLabel()`**).

7.2.1. Libellés des erreurs locales

Les erreurs locales sont des instances de **`LocalException`**.

Les clés permettant de retrouver les libellés des erreurs locales correspondent aux noms des constantes définies sur la classe **`com.ibm.vap.generic.LocalException`**. Les clés représentent les différents type d'erreurs locales et sont préfixées par **`LOCAL_`** afin d'éviter toute confusion avec des codes d'erreur définis par l'utilisateur dans le fichier des libellés d'erreur.

Exemple : *La clé permettant de retrouver le libellé associé à une exception locale de type* ***`INVALID_INSTANCE`*** *sera* ***`LOCAL_INVALID_INSTANCE`***.

7.2.2. Libellés locaux des messages d'erreurs reçus du serveur

Les messages d'erreurs reçus du serveur comportent deux informations : une clé et un libellé.

La clé du message d'erreur va être interprétée afin de déterminer la clé de stockage du libellé local associé à l'erreur.

La structure de la clé du message d'erreur est décrite dans le *Guide Utilisateur Volume III – Clients Graphiques*.

Dans le cas d'erreur système, la clé d'accès au libellé local correspond au préfixe **`SYSTEM_`** suivi des caractères compris entre les colonnes 14 et 19 s'ils ont une valeur significative (différente de blanc) ou bien suivi des caractères compris entre les colonnes 10 et 13, si la valeur n'est pas significative.

Dans le cas d'erreurs serveur (erreurs utilisateur, par exemple), la clé d'accès au libellé local correspond aux caractères compris entre les colonnes 14 et 19 s'ils ont une valeur significative et si le caractère situé en colonne 20 est blanc. Si le caractère situé en colonne 20 est 2 ou 5, la clé d'erreur est respectivement **`REQUIRED`** et **`VALUE`**.

Si les caractères compris entre les colonnes 14 et 19 n'ont pas une valeur significative, la clé d'accès au libellé local correspond aux caractères compris entre les colonnes 22 et 25.

7.2.3. Libellés pour les erreurs serveur et système

Le libellé des exceptions de type **com.ibm.vap.generic.ServerException** et des erreurs de type **com.ibm.vap.generic.SystemError** sont respectivement accessibles par les clés **VAP_SERVER_EXCEPTION** et **VAP_SYSTEM_ERROR**.

7.2.4. Exemple de fichier de libellés d'erreurs

```
# Ce fichier définit les libellés d'erreur locales et serveurs pour le
produit VisualAgePacbase for Java.
# Les libellés stockés dans ce fichier sont définis selon le formalisme des
patterns de java.text.MessageFormat.
# Les arguments possibles pour ces libellés sont :
# {0} = bibliothèque
# {1} = nom du serveur
# {2} = code de la vue
# {3} = data id
# {4} = nom de la rubrique
# {5} = valeur de la rubrique
# {6} = libellé technique (message technique renseigné par le cache local
ou reçu du serveur)
# Remarque : Ces arguments sont renseignés selon le contexte de l'erreur.
Ils peuvent donc ne pas avoir de valeurs si l'argument n'a pas de sens dans
le contexte.

# Libellés Local Exception
LOCAL_PARENT_INSTANCE_MISSING = Instance parente absente (data id: {3})
LOCAL_CURRENT_INSTANCE_MISSING = Instance courante absente
LOCAL_SERVER_UPDATE_REQUIRED = Mise à jour serveur nécessaire (data id:
{3})
LOCAL_UNKNOWN_INSTANCE = L'instance est inconnue (data id: {3})
LOCAL_INVALID_INSTANCE = L'instance est invalide
LOCAL_INSTANCE_NOT_LOCKED = L'instance n'est pas verrouillée (data id: {3})
LOCAL_INVALID_CREATION = Création invalide (data id: {3})
LOCAL_INVALID_CHANGE = Modification invalide (data id: {3})
LOCAL_INVALID_DELETION = Suppression invalide (data id: {3})
LOCAL_INVALID_INITIALIZATION = Initialisation invalide (data id: {3})
LOCAL_CARDINALITY_VIOLATION = Cardinalitées non respectées {6} (data id:
{3})
LOCAL_INSTANCE_ALREADY_LOCKED = L'instance est déjà verrouillée (data id:
{3})
LOCAL_CURRENT_USER_INSTANCE_MISSING = Instance utilisateur courante absente
LOCAL_REFERING_INSTANCE_MISSING = Instance référençante absente
LOCAL_ASYNCHRONOUS_VIOLATION = Erreur sur utilisation asynchrone ({6})
LOCAL_UNKNOWN_CONTEXT = Contexte inconnu
LOCAL_VALUE_REQUIRED = Rubrique obligatoire: {4} (data id: {3})
LOCAL_VALUE_ERROR = Erreur de valeur: {4} (valeur: {5}, data id: {3})
LOCAL_LENGTH_ERROR = Erreur de longueur sur la rubrique: {4} (valeur: {5},
data id: {3})
LOCAL_SUBSCHEMA_ERROR = La rubrique {4} n'appartient pas au sous-schema
(valeur: {5}, data id: {3})
LOCAL_FOLDER_USER_CONTEXT_LENGTH_ERROR = Erreur de longueur sur la rubrique
{4} du buffer utilisateur du dossier (value: {5})
```

```
LOCAL_REFERENCE_USER_CONTEXT_LENGTH_ERROR = Erreur de longueur sur la
rubrique {4} d’un buffer utilisateur d’un noeud référence (value: {5})

# Libellés Erreur Service Serveur
REQUIRED = Valeur obligatoire pour la rubrique {4} (bib: {0}, ser: {1},
vue: {2})
VALUE = Erreur de valeur pour la rubrique {4} (bib: {0}, ser: {1}, vue:
{2})
DUPL = Création à tort (bib: {0}, ser: {1}, vue: {2})
NFND = Modif/Annulation à tort (bib: {0}, ser: {1}, vue: {2})
LOCKED = Instance déjà verrouillée (bib: {0}, ser: {1})
NTLOCK = Instance non verrouillée (bib: {0}, ser: {1})

# Libellés Erreur Système
SYSTEM_STRU = Erreur de structure sur la Vue Logique (bib: {0}, ser: {1})
SYSTEM_VERS = Erreur de version (bib: {0}, ser: {1})
SYSTEM_VIEW = Vue inconnue (bib: {0}, ser: {1})
SYSTEM_SERV = Service inconnu (bib: {0}, ser: {1})
SYSTEM_LTH = Longueur de vue erronée (bib: {0}, ser: {1})
SYSTEM_LSRV = Longueur du message reçu erronée (bib: {0}, ser: {1})
SYSTEM_NUVE = Erreur de version dans les composants (bib: {0}, ser: {1})
SYSTEM_PCVLTH = Longueur de message erronné (bib: {0}, ser: {1})
SYSTEM_MISPCV = Déphasage composants
SYSTEM_ACCESS = Erreur d’accès aux données {6} (bib: {0}, ser: {1}, vue:
{2})
SYSTEM_LKABSC = Verrouillage non implémenté (bib: {0}, ser: {1})
SYSTEM_WF00 = Erreur sur le fichier de travail ou Erreur sur la connexion à
la base de données (erreur : {6})
SYSTEM_TAND = Erreur Pathsend {6} (bib: {0}, ser: {1})
SYSTEM_PILO = Enregistrement pilote non trouvé lors du traitement du buffer
utilisateur (bib: {0}, ser: {1})
SYSTEM_EXT1 = Méthode d’extraction : erreur syntaxe PCV ou trop long (bib:
{0}, ser: {1})
SYSTEM_EXT2 = Méthode d’extraction : inconnue du gestionnaire de dossier
(bib: {0}, ser: {1})
SYSTEM_USR1 = Service utilisateur absent (bib: {0}, ser: {1})
SYSTEM_USR2 = Service utilisateur trop long (bib: {0}, ser: {1})
SYSTEM_USR3 = Service utilisateur inconnu du gestionnaire de dossier (bib:
{0}, ser: {1})

# Libellés Exception internes
VAP_SERVER_EXCEPTION = Une exception serveur s’est produite.
VAP_SYSTEM_ERROR = Une erreur système s’est produite.
```

8. INDEX

Actions COM

Actions Smalltalk

Attributs COM

Attributs Smalltalk

Classes

Evénements COM

Evénements Java

Evénements Smalltalk

Méthodes Java

Propriétés Java