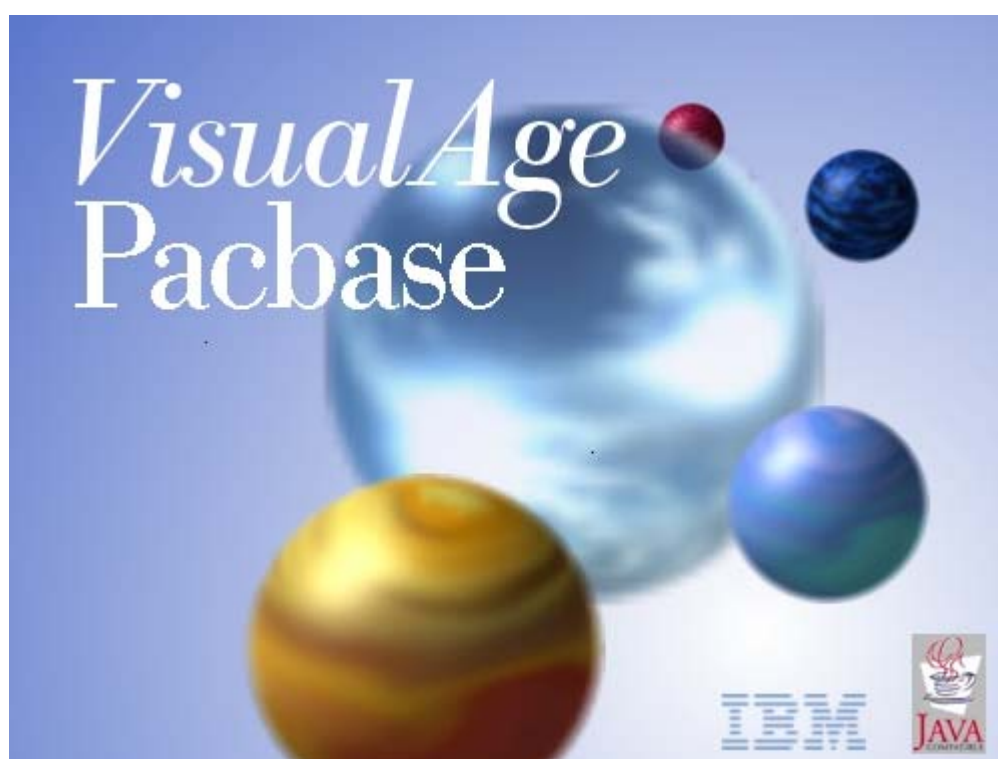


VisualAge Pacbase



**Applications eBusiness & Pacbench C/S**  
**Interface de programmation des Proxies**

*Version 3.5*



Remarque

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

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/awdtools/vapacbase/productinfo\\_f.htm](http://www.ibm.com/software/awdtools/vapacbase/productinfo_f.htm)

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

**Première Edition (Juin 2004)**

La présente édition s'applique à :

VisualAge Pacbase Version 3.5

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/awdtools/vapacbase/support.htm>

ou en nous adressant un courrier à :

IBM Paris Laboratory

Support VisualAge Pacbase

1 place J.B. Clément

93881 Noisy-le-Grand Cedex, France

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

© Copyright International Business Machines Corporation 1983,2004. Tous droits réservés.

---

# Sommaire

La *Table des Matières* détaillée est proposée dans les pages suivantes.

<b>Notices</b> .....	<b>ix</b>	Actions exécutées sur un serveur distant.....	68
<b>Marques</b> .....	<b>xi</b>	Externalisation de la gestion des requêtes.....	86
<b>Préambule</b> .....	<b>1</b>	Manipulation de flots XML.....	87
<b>Chapitre 1. Classes</b> .....	<b>3</b>	Gestion du contexte d'une proxy.....	90
Classes de données.....	3	<b>Chapitre 4. Événements</b> .....	<b>93</b>
Classe ProxyLv : Schémas d'héritage.....	5	Gestion de la pagination.....	93
Classes de manipulations de flots XML.....	5	Gestion des lectures unitaires.....	94
<b>Chapitre 2. Attributs</b> .....	<b>7</b>	Gestion des sélections simultanées.....	94
Gestion des sélections.....	7	Gestion du verrouillage logique.....	95
Gestion des mises à jour.....	11	Gestion des instances dépendantes.....	96
Contrôle du flux des échanges.....	13	<b>Chapitre 5. Interface publique de</b>	
Container d'instances de Vue Logique.....	15	<b>manipulation des Rubriques</b> .....	<b>97</b>
Gestion des Services Utilisateur.....	24	Gestion du contenu de la Rubrique.....	97
Gestion des verrouillages logiques.....	25	Gestion des codes des valeurs permises.....	97
Gestion des messages de retour de sélection.....	26	Gestion des libellés des valeurs permises.....	98
Gestion des événements.....	27	Gestion de la validité du contenu d'une Rubrique.....	98
Gestion des informations contextuelles.....	28	Accès aux caractéristiques d'une Rubrique.....	99
Compteurs disponibles.....	29	Initialisation de l'ensemble des valeurs des Rubriques	
Gestion des communications.....	31	composant une Vue Logique.....	99
Gestion des conversations asynchrones.....	40	Gestion de la présence d'une Rubrique.....	100
Temps de conversation.....	42	Gestion du contrôle d'une Rubrique.....	100
Gestion des sous-schémas.....	43	Gestion d'appartenance à un sous-schéma.....	101
Gestion des requêtes externes.....	45	Gestion d'appartenance à une méthode d'extraction.....	101
Utilisation d'une JTable.....	46	<b>Chapitre 6. Gestion des Erreurs</b> .....	<b>103</b>
<b>Chapitre 3. Actions</b> .....	<b>49</b>	Gestion des erreurs pour la cible Java.....	103
Actions exécutées localement.....	49	Gestion des erreurs pour la cible COM.....	108
		<b>Index</b> .....	<b>115</b>



# Table des Matières

<b>Notices</b> .....	<b>ix</b>
<b>Marques</b> .....	<b>xi</b>
<b>Préambule</b> .....	<b>1</b>
<b>Chapitre 1. Classes</b> .....	<b>3</b>
Classes de données.....	3
Schémas d'héritage.....	3
Java et COM.....	3
Classe DataDescription.....	3
Classe SelectionCriteria .....	4
Classe DataDescriptionUpdate .....	4
Classe UserContext.....	4
Classe ProxyLv : Schémas d'héritage .....	5
Java et COM.....	5
Classes de manipulations de flots XML .....	5
Schéma d'héritage.....	5
Java et COM.....	5
Classes génériques.....	6
Classes générées.....	6
<b>Chapitre 2. Attributs</b> .....	<b>7</b>
Gestion des sélections.....	7
Critères de sélection .....	7
Liste des méthodes d'extraction disponibles .....	8
Méthode d'extraction à exécuter .....	9
Mode de gestion de la collection.....	10
Gestion des mises à jour .....	11
Mise en œuvre des contrôles des données sur le serveur .....	11
Rafraîchissement des données sur le serveur .....	12
Contrôle du flux des échanges.....	13
Limitation du nombre d'instances échangées.....	13
Nombre d'instances échangées illimité.....	14
Container d'instances de Vue Logique .....	15
Présentation d'une liste d'instances.....	15
Sélection du tri local ou serveur sur une liste d'instances.....	16
Critère local de tri d'une liste d'instances.....	17
Présentation d'une instance .....	18
Présentation des Dossiers modifiés .....	19
Présentation des instances modifiées .....	20
Présentation d'instances destinées à un Service Utilisateur.....	21
Présentation d'instances rendues par un Service Utilisateur.....	22
Présentation d'une instance liée à un Service Utilisateur.....	23
Gestion des Services Utilisateur .....	24
Liste des Services Utilisateur disponibles.....	24
Services Utilisateur à exécuter.....	24
Gestion des verrouillages logiques .....	25
Identifiant de verrouillage d'un Dossier .....	25
Gestion des messages de retour de sélection.....	26
Libellé du message.....	26
Clé du message .....	26
Gestion des événements.....	27
Liste des événements de la dernière action serveur .....	27
Gestion des informations contextuelles .....	28
Informations contextuelles .....	28
Informations contextuelles liées aux nœuds références.....	28
Compteurs disponibles.....	29
Nombre total d'instances locales .....	29
Nombre total de services de mise à jour.....	29
Nombre de services de mise à jour associés à un nœud .....	30
Gestion des communications.....	31
Liste des plates-formes disponibles .....	31
Plate-forme sélectionnée pour l'exécution d'une requête.....	32
Code utilisateur de connexion à la plate-forme .....	32
Mot de passe de connexion à la plate-forme .....	33
Nom de la machine abritant la gateway Java/Com VisualAge Pacbase .....	33
Port IP associé au gestionnaire de communication .....	34
Initialisation de l'adresse du fichier des plates-formes.....	34
Choix de l'adaptateur de communication.....	35
Gestion des paramètres de communication.....	36
Gestion des conversations asynchrones .....	40
Détermination du type de conversation.....	40
Dernier identifiant de conversation asynchrone.....	40
Nombre maximum de réponses en attente.....	41
Nombre de réponses en attente .....	41
Temps de conversation.....	42
Temps de communication.....	42
Temps d'exécution du traitement serveur .....	42
Gestion des sous-schémas.....	43
Liste des sous-schémas disponibles .....	43
Sous-schéma à prendre en compte.....	44
Gestion des requêtes externes .....	45
Accès et affectation d'une requête .....	45
Utilisation d'une JTable .....	46
Affichage de la collection des instances dans une JTable .....	46
Affichage des Dossiers modifiés dans une JTable .....	46
Affichage des instances modifiées dans une JTable .....	47
Affichage de la collection d'instances destinées ou rendues par un Service Utilisateur dans une JTable .....	47
<b>Chapitre 3. Actions</b> .....	<b>49</b>
Actions exécutées localement .....	49
Mises à jour .....	49
Création d'une instance de Vue Logique.....	49
Modification d'une instance de Vue Logique .....	50
Suppression d'une instance de Vue Logique .....	51

Annulation des mises à jour.....	52	Contrôle de présence d'instances dépendantes ...	82
Annulation des mises à jour d'un Dossier.....	52	Gestion des Services Utilisateur.....	83
Annulation des mises à jour de toutes les instances du Dossier.....	53	Exécution de Services Utilisateur .....	83
Annulation des mises à jour d'une instance de nœud.....	54	Gestion des conversations asynchrones .....	84
Annulation des mises à jour de toutes les instances d'un nœud .....	55	Récupération différée d'une réponse.....	84
Gestion des Services Utilisateur .....	56	Contrôle de la validité d'un identifiant de message.....	85
Affectation d'une instance à un Service Utilisateur.....	56	Gestion des sous-schémas.....	85
Modification d'une instance affectée.....	57	Test de communication avec le serveur.....	86
Suppression d'une instance affectée.....	58	Externalisation de la gestion des requêtes .....	86
Navigation locale dans les Dossiers .....	59	Création d'une requête .....	86
Sélection courante d'une instance dans un Dossier.....	59	Exécution sur le serveur des actions de la requête .....	87
Sélection d'une instance à partir d'un index.....	60	Abandon des actions de la requête.....	87
Sélection d'une instance associée à un Service Utilisateur .....	60	Manipulation de flots XML.....	87
Remise en place de la sélection courante.....	61	Etat d'une instance de vue logique .....	88
Initialisations diverses .....	61	Obtention d'un flot XML.....	88
Initialisation de la collection .....	61	Mise à jour à partir d'un flot XML.....	89
Initialisation des méthodes d'extraction.....	62	Gestion du contexte d'une proxy.....	90
Initialisation des Services Utilisateur .....	62	Initialisation du contexte d'une proxy.....	90
Initialisation du container présentation d'instances destinées à un Service Utilisateur .....	63	Initialisation du cache local .....	90
Initialisation de l'option de rafraîchissement des mises à jour .....	63	Récupération de la proxy .....	91
Initialisation des critères de sélection.....	64	<b>Chapitre 4. Evénements .....</b>	<b>93</b>
Peuplement du cache local sans accès serveur .....	64	Gestion de la pagination.....	93
Gestion des instances référencées .....	65	Signal de récupération de la dernière page d'une collection.....	93
Affectation d'une instance référencée .....	65	Signal de récupération de la première page d'une collection.....	93
Récupération des contextes de génération des Proxies.....	66	Signal de présence d'au moins une page suivante .....	93
Contexte de génération d'un Dossier.....	66	Signal de présence d'au moins une page précédente.....	94
Contexte de génération d'un nœud.....	67	Gestion des lectures unitaires.....	94
Gestion des sous-schémas .....	67	Signal de lecture d'un enregistrement non trouvé .....	94
Aucune sélection de sous-schéma .....	67	Gestion des sélections simultanées.....	94
Actions exécutées sur un serveur distant .....	68	Signal de non participation à une lecture simultanée.....	94
Sélection sur un nœud .....	68	Gestion du verrouillage logique .....	95
Sélection d'un ensemble d'instances .....	68	Signal de verrouillage logique effectué .....	95
Lecture d'une instance avec ou sans verrouillage logique.....	69	Signal de verrouillage logique infructueux .....	95
Lecture d'instances à partir d'identifiants.....	70	Gestion des instances dépendantes.....	96
Sélection simultanée sur plusieurs nœuds avec ou sans verrouillage.....	71	Signal de présence d'au moins une instance dépendante.....	96
Lecture d'une instance et de sa hiérarchie immédiate .....	71	Signal d'absence d'instances dépendantes .....	96
Lecture d'une instance et de sa hiérarchie complète .....	72	<b>Chapitre 5. Interface publique de manipulation des Rubriques.....</b>	<b>97</b>
Lecture de la hiérarchie immédiate d'une instance courante.....	74	Gestion du contenu de la Rubrique.....	97
Lecture de la hiérarchie complète d'une instance courante.....	74	Gestion des codes des valeurs permises.....	97
Lecture de la hiérarchie immédiate d'une instance par anticipation .....	75	Gestion des libellés des valeurs permises .....	98
Lecture de la hiérarchie complète d'une instance par anticipation .....	75	Gestion de la validité du contenu d'une Rubrique .....	98
Gestion de la pagination.....	76	Accès aux caractéristiques d'une Rubrique .....	99
Lecture des instances de la page suivante.....	76	Initialisation de l'ensemble des valeurs des Rubriques composant une Vue Logique .....	99
Lecture des instances de la page précédente .....	77	Gestion de la présence d'une Rubrique .....	100
Emission des mises à jour.....	79	Gestion du contrôle d'une Rubrique.....	100
Emission des mises à jour locales sur le serveur.....	79	Gestion d'appartenance à un sous-schéma .....	101
Gestion du verrouillage logique.....	80		
Verrouillage logique d'une instance courante.....	80		
Déverrouillage logique d'une instance courante .....	81		
Gestion des instances dépendantes .....	82		

Gestion d'appartenance à une méthode d'extraction ..... 101

## **Chapitre 6. Gestion des Erreurs .....103**

Gestion des erreurs pour la cible Java .....	103
Classes relatives à la gestion des erreurs .....	104
Erreurs de communication.....	104
Erreurs système.....	104
Erreurs locales .....	104
Erreurs serveur.....	105
Message d'erreur reçu du composant Serveur105	
Message d'erreur reçu du composant Serveur	
sur mise à jour.....	105
Personnalisation des libellés des erreurs .....	106
Libellé des exceptions locales .....	106
Libellés locaux des messages d'erreur reçus du	
composant serveur .....	106
Libellés pour les erreurs serveur et système .....	107
Exemple de fichier de libellés d'erreur .....	107
Gestion des erreurs pour la cible COM.....	108
Méthodes d'accès aux erreurs .....	108
Attributs de VapError .....	109
Gestion du type de l'erreur .....	109

Gestion de l'action provoquant l'erreur.....	109
Gestion de la clef d'erreur.....	109
Gestion du libellé de l'erreur .....	109
Gestion de la gravité de l'erreur .....	109
Evénements liés aux erreurs.....	110
Signal de non détection d'erreur .....	110
Signal de récupération d'une erreur locale.....	110
Signal de récupération d'une erreur serveur.....	110
Signal de récupération d'une erreur système.....	110
Signal de récupération d'une erreur de	
communication .....	111
Personnalisation des libellés des erreurs.....	111
Règles de nommages des fichiers de libellés	
d'erreur.....	111
Syntaxe des fichiers de libellés d'erreur .....	111
Libellé des exceptions locales.....	111
Libellés locaux des messages d'erreur reçus du	
composant serveur .....	112
Libellés globaux des erreurs serveur et système	
Exemple de fichier de libellés d'erreur .....	112

**Index..... 115**





---

## Notices

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

1 place J.B. Clément

93881 Noisy-le-Grand Cedex

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.



---

## Préambule

Ce manuel décrit exhaustivement l'interface publique des composants générés pour les Clients graphiques des applications eBusiness en fonction des environnements cibles ; Java 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 Élémentaire 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 Élémentaire qui lui est associé.

### Organisation du manuel

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

- Le premier chapitre, *Chapitre 1. Classes*, page 3, présente les classes de l'interface publique, avec les arbres d'héritage.
- Le second chapitre, *Chapitre 2. Attributs*, page 7, donne la liste de tous les types d'attributs sur les différentes plates-formes. Pour chaque attribut, le type, le code interne, le nom d'utilisation et le **get/set** sont donnés.
- Le troisième chapitre, *Chapitre 3. Actions*, page 49, décrit les actions (ou méthodes pour la plate-forme Java) – qu'elles soient locales ou distantes – avec la signature et le nom d'utilisation.
- Le quatrième chapitre, *Chapitre 4. Événements*, page 93, documente les événements et donne leur code.
- Le cinquième chapitre, *Chapitre 5. Interface publique de manipulation des Rubriques*, page 97, documente l'API de manipulation des Rubriques.
- Le sixième chapitre, *Chapitre 6. Gestion des Erreurs*, page 103, traite la gestion des erreurs pour les cibles Java et COM..

### Prérequis et autres lectures

Avant de lire ce volume, vous devez impérativement connaître les grands principes du module eBusiness de VisualAge Pacbase. Ils sont supposés connus dans le présent volume. Consultez le manuel *Applications eBusiness & Pacbench C/S - Concepts et Architectures*.

Si vous débutez dans ce type de développement, nous vous conseillons la lecture du manuel *Applications eBusiness & Pacbench C/S - Présentation Graphique*. Ce guide vous aidera dans le développement d'un composant Client graphique au travers de nombreux exemples.

## Conventions typographiques

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.



Renvoi, *en italiques*, à un autre emplacement dans le présent volume ou à un autre manuel. Ces renvois sont des liens hypertexte (positionnez le curseur de la souris, avec un double-clic vous afficherez la cible du renvoi).

Attention, pour un renvoi à un autre manuel, vous ouvrez la page d'accueil de la Documentation diffusée sur le site Internet de VisualAge Pacbase, repérez le manuel en question et double-cliquez.



Précaution à prendre (manipulation risquée ou irréversible...).

---

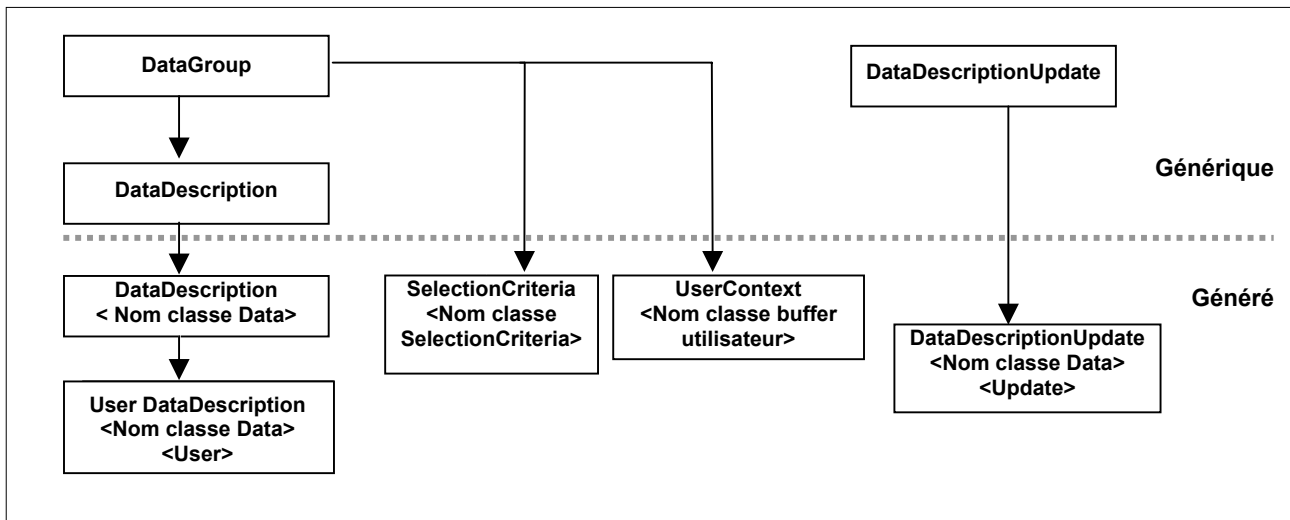
# Chapitre 1. Classes

---

## Classes de données

### Schémas d'héritage

Java et COM



- Depuis VisualAge Pacbase 3.0, le nommage des classes DataDescription, SelectionCriteria et UserContext se fait lors de la composition des Dossiers et de l'Application eBusiness.

La génération des nœuds références étant optionnelle, le nommage est dynamique et suit la règle suivante :

<Nom de la classe définie><Nom du Dossier>

- Si la génération est obtenue à partir de Dossiers définis en VA Pac 2.5, le nommage des classes suit les règles suivantes :

DataDescription	:	<VueLogique><Data>
User DataDescription	:	<VueLogique><UserData>
SelectionCriteria	:	<VueLogique>< SelectionCriteria>
UserContext	:	<VueLogique>< UserContext>
DataDescriptionUpdate	:	<VueLogique><DataUpdate>

### Classe DataDescription

Cette classe est générée pour chaque 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.

### 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.

### Classe DataDescriptionUpdate

Cette classe est générée pour les nœuds racines et dépendants modifiables. 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 associée au nœud a été supprimée localement.
- **Read** : Au moins une des instances dépendantes dans le Dossier a été mise à jour localement.

### Classe UserContext

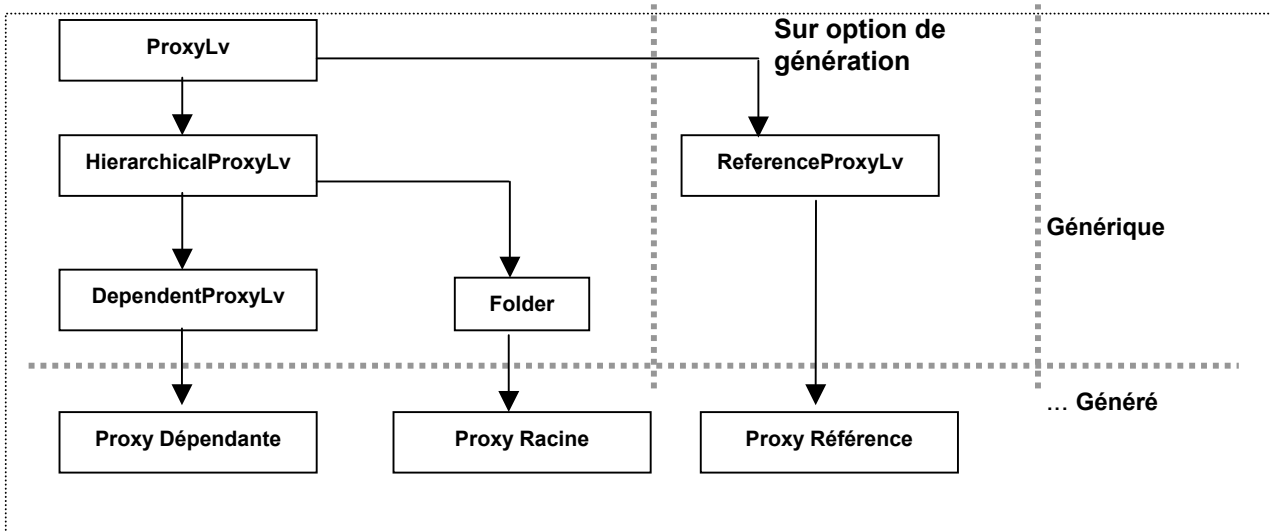
Cette classe est générée pour chaque Application eBusiness dans laquelle a été défini un buffer utilisateur. Elle représente la structure de ce buffer en définissant un attribut par Rubrique.

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.



## Classe ProxyLv : Schémas d'héritage

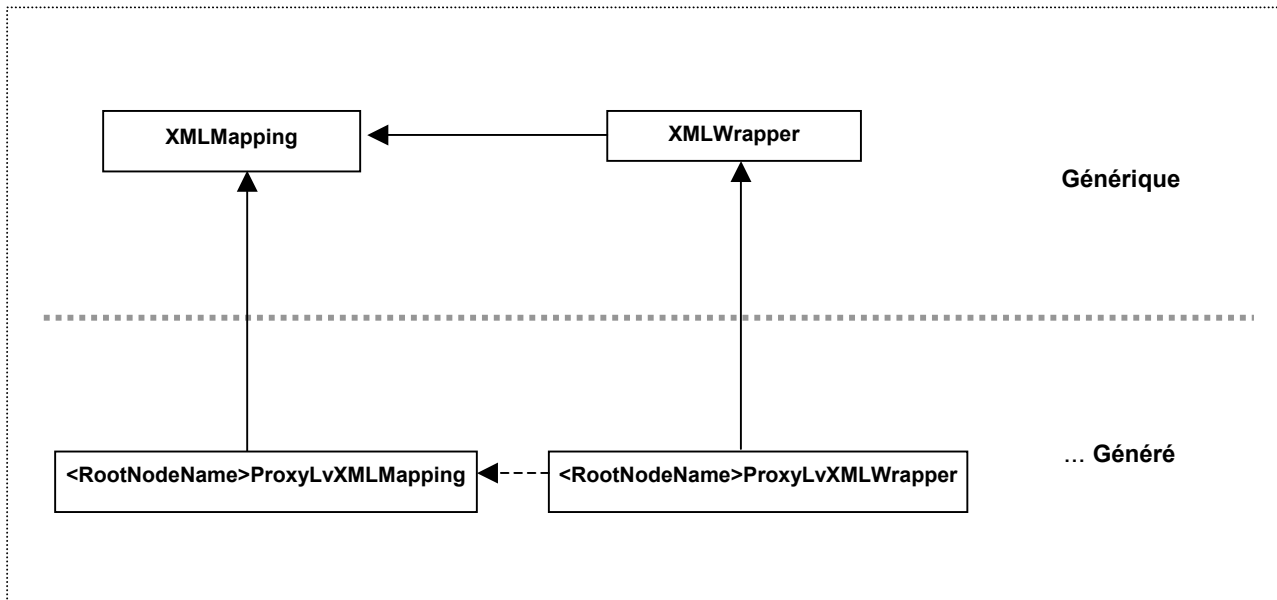
Java et COM



## Classes de manipulations de flots XML

Schéma d'héritage

Java et COM



A l'issue de la génération, un schéma XML est également constitué pour un dossier ou une vue de dossier (codifié <FolderName>.xsd ou <FolderName-ViewName>.xsd). Ce schéma, conforme aux recommandations du W3C contient la description des données.

## **Classes génériques**

Les classes XMLMapping et XMLWrapper définissent l'api générique de manipulation des flots XML.

## **Classes générées**

La classe de mapping XML est une classe singleton de type XMLMapping. Elle décrit les éléments du mapping spécifique à un dossier ou à une vue de dossier.

La classe wrapper XML est de type XMLWrapper à laquelle on associe l'instance de classe XMLMapping générée. Elle offre des méthodes permettant pour chaque nœud de masquer les éléments de la demande et elle est générée pour un dossier ou une vue de dossier.

---

## Chapitre 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.

---

### Gestion des sélections

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

##### COM

- Type `{Nom de classe SelectionCriteria générée}`
- Code interne `selectionCriteria`
- get/set C++ `public LPDISPATCH getSelectionCriteria() / set non disponible`

## Liste des méthodes d'extraction disponibles

### Description

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

Il est disponible sur les nœuds racines, dépendants et références lorsqu'au moins une méthode d'extraction est définie dans le Composant Élémentaire 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

### COM

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

- Code interne `getExtractMethodCodes`
- Signature `public VapCollection getExtractMethodCodes ()`  
Disponible également avec une API de parcours de collection
- Nb d'éléments `public long getExtractMethodCodesCount ()`
- Élément `public char* getExtractMethodCodesElementAt (long i)`

## 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 racines, dépendants et références lorsqu'au moins une méthode d'extraction est définie dans le Composant Élémentaire 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)
```

### COM

- Type `BSTR`
- Code interne `extractMethodCode`
- get/set C++ 

```
public BSTR getExtractMethodCode() /  
public void setExtractMethodCode(LPCTSTR s)
```

## 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 sont 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 références est toujours de type **extend**.

Cet attribut est systématiquement disponible sur les nœuds racines, dépendants ou références.

Il est accessible en lecture ou en écriture.

### Java

• Type	<b>Boolean</b>
• Code interne	<b>manualCollectionReset</b>
• Nom d'utilisation	<b>Manual Collection Reset</b>
• get/set	<b>public Boolean isManualCollectionReset() public void setManualCollectionReset(boolean b)</b>

### COM

• Type	<b>Boolean</b>
• Code interne	<b>manualCollectionReset</b>
• get/set C++	<b>public BOOL getManualCollectionReset() public void setManualCollectionReset(BOOL b)</b>

---

## Gestion des mises à jour

### Mise en œuvre des contrôles des données sur le serveur

#### Description

Cet attribut permet de déclencher les contrôles 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 racines et dépendants lorsque la Vue Logique associée est utilisée en mise à jour dans son Composant Élémentaire et que l'option **CHECKSER** de ce Composant Élémentaire 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)**

#### COM

- Type **Boolean**
- Code interne **serverCheckOption**
- get/set C++ **public BOOL getServerCheckOption()  
public void setServerCheckOption(BOOL b)**

## 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 Élémentaire. 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 racines et dépendants lorsque la Vue Logique associée est utilisée en mise à jour dans son Composant Élémentaire.

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)**

### COM

- Type **Boolean**
- Code interne **refreshOption**
- get/set C++ **public BOOL getRefreshOption()  
public void setRefreshOption(BOOL b)**



---

## Contrôle du flux des échanges

### 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 Élémentaire 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 sélections simultanées sur plusieurs nœuds ne propagent pas de requête de lecture sur le nœud concerné.

Il est disponible sur les nœuds racines ou références 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)
```

#### COM

- Type `Integer`
- Code interne `maxNumberOfRequestedInstances`
- get/set C++ 

```
public long getMaxNumberOfRequestedInstances ()
public void setMaxNumberOfRequestedInstances (Long i)
```

## Nombre d'instances échangées illimité

### Description

Cet attribut permet de récupérer la totalité des instances contenues 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 Élémentaire.

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 racines ou références 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)**

### COM

- Type **Boolean**
- Code interne **globalSelection**
- get/set C++ **public BOOL getGlobalSelection()  
public void setGlobalSelection(BOOL b)**

---

## Container d'instances de Vue Logique

### 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 récupérés en local.

Pour un nœud dépendant, la collection des instances exposées dépend de l'instance de Vue Logique contenue dans l'attribut **detail** 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 racines ou références 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

#### COM

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

- Code interne `getRows`
- Signature `public UcpCollection getRows()`  
Disponible également avec une API de parcours de collection
- Nb d'éléments `public Long getRowCount()`
- Élément `public {Nom de classe DataDescription générée} getRowsElementAt(Long i)`

## 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 15, 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 racines ou références 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 **Local Sort**
- get/set **public boolean getLocalSort() / public setLocalSort(boolean)**

#### COM

- Type **Boolean**
- Code interne **localSort**
- get/set **public Boolean isLocalSort()  
public void setLocalSort(BOOL a)**

## 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 15.

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 racines ou références 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
- Code interne
- Nom d'utilisation -
- get/set

```
com.ibm.vap.generic.Comparator
```

```
dataComparator
```

```
public com.ibm.vap.generic.Comparator getDataComparator()
```

```
public void setDataComparator(com.ibm.vap.generic.Comparator c)
```

### COM

Non disponible

## 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 racines, dépendants et références.

Il est accessible en lecture ou en écriture.

#### Java

- Type `{Nom de classe DataDescription générée}`
- Code interne `detail`
- Nom d'utilisation `Detail`
- get/set `public {Nom de classe DataDescription générée} detail() / set non disponible`

#### COM

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

## 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 modifications associés à l'instance de Dossier modifiée.
- Le statut de modification pouvant 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 Élémentaires 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

### COM

- Nb d'éléments Disponible avec une API de parcours de collection  
`public Long getUpdatedFoldersCount()`
- Élément `public {Nom de classe DataUpdate générée}  
getUpdatedFoldersElementAt(Long i)`

## Présentation des instances modifiées

### Description

Cet attribut permet d'exposer la liste des instances du nœud 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 nœud modifié :

- L'instance de Vue Logique du nœud.
- Le nombre de services de modifications associés à l'instance modifiée.
- Le statut de modification pouvant 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 Élémentaire associé au nœud 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

### COM

- Nb d'éléments Disponible avec une API de parcours de collection  
`public Long getUpdatedInstancesCount ()`
- Élément `public {Nom de classe DataUpdate générée}`  
`getUpdatedInstancesElementAt (Long i)`



## 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 15.

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 racines ou dépendants lorsqu'au moins un Service Utilisateur est défini dans le Composant Élémentaire 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

### COM

- Nb d'éléments Disponible avec une API de parcours de collection  
`public Long getUserInputRowsCount ()`
- Élément `public {Nom de classe UserDataDescription générée}`  
`getUserInputRowsElementAt (Long i)`

## 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 15.

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 racines ou dépendants lorsqu'au moins un Service Utilisateur est défini dans le Composant Élémentaire 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

### COM

- Nb d'éléments Disponible avec une API de parcours de collection  
`public Long getUserOutputRowCount ()`
- Élément `public {Nom de classe UserDataDescription générée}`  
`getUserOutputRowsElementAt (Long i)`

## 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 racines ou dépendants lorsqu'au moins un Service Utilisateur est défini dans le Composant Élémentaire qui gère la Vue Logique associée au nœud.

Il est accessible en lecture uniquement.

### Java

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

### COM

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

---

## Gestion des Services Utilisateur

### 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 racines ou dépendants lorsqu'au moins un Service Utilisateur est défini dans le Composant Élémentaire 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

#### COM

- Nb d'éléments Disponible avec une API de parcours de collection  
`public Long getUserServiceCodesCount ()`
- Élément `public String getUserServiceCodesElementAt (Long i)`

### 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 racines ou dépendants lorsqu'au moins un Service Utilisateur est défini dans le Composant Élémentaire 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)`

#### COM

- Type `String`
- Code interne `userServiceCode`
- get/set C++ `public BSTR getUserServiceCode ()`  
`public void setUserServiceCode (LPCTSTR s)`

---

## Gestion des verrouillages logiques

### 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`

#### COM

- Type `String`
- Code interne `lockTimestamp`
- get/set C++ `public BSTR getLockTimestamp() / set non disponible`

---

## Gestion des messages de retour de sélection

### 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 nœuds.

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

#### COM

- Type `String`
- Code interne `accessInfoLabel`
- get/set C++ `public BSTR getAccessInfoLabel ()` / set non disponible

### 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 nœuds.

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

#### COM

- Type `String`
- Code interne `accessInfoKey`
- get/set C++ `public BSTR getAccessInfoKey ()`  
set non disponible

---

## Gestion des événements

### 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

#### 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
- Élément

```
public Long getServerEventsCount ()
```

```
public String popServerEvent ()
```

---

## Gestion des informations contextuelles

### 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 de l'application eBusiness. Il est accessible en lecture uniquement.

#### Java

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

#### COM

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

### Informations contextuelles liées aux nœuds références

#### 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 Élémentaire gérant le nœud référence. Il est accessible en lecture uniquement.

#### Java

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

#### COM

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



---

## Compteurs disponibles

### 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

#### 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 Long 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 Vues Logiques 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 Élémentaire 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

#### 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 Long getFolderUpdatedInstancesCount()`  
set non disponible

## 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 Élémentaire la gérant.

Il est accessible en lecture uniquement.

### Java

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

### COM

- get/set C++ Cette information n'est pas référencée en tant qu'attribut dans l'API de la Proxy. Elle est disponible en exécutant la méthode suivante :  
`public long getNodeUpdatedInstancesCount()`  
set non disponible

---

## Gestion des communications

### Liste des plates-formes disponibles

#### Description

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

Lors de l'utilisation 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 localisations associées au Dossier.

Cet attribut est systématiquement disponible sur le nœud racine.

Il est accessible en lecture ou en écriture.

#### Java

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

#### COM

- Nb d'éléments Disponible avec une API de parcours de collection  
`public Long getLocationsCount()`
- Élément `public String getLocationsElementAt(Long i)`

## Plate-forme 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 localisation. 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 61.

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)`

#### COM

- Type `String`
- Code interne `location`
- get/set C++ `public BSTR getLocation() /`  
`public void setLocation(LPCTSTR l)`

## Code utilisateur de connexion à la plate-forme

### Description

Cet attribut contient le code utilisateur de connexion à la plate-forme 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)`

#### COM

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

## Mot de passe de connexion à la plate-forme

### Description

Cet attribut contient le mot de passe du code utilisateur de connexion à la plate-forme 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)**

#### COM

- Type **String**
- Code interne **password**
- get/set C++  
get non disponible / **public void setPassword(LPCTSTR p)**

## 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 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)**

#### COM

- Type **String**
- Code interne **host**
- get/set C++  
**public BSTR getHost() / public void setHost(LPCTSTR h)**

## 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 Elémentaires.

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)`

#### COM

- Type `Long`
- Code interne `port`
- get/set C++ `public Long getPort() / public void setPort(Long p)`

## Initialisation de l'adresse du fichier des plates-formes

### Description

Cet attribut contient le chemin complet du fichier des plates-formes utilisé par le gestionnaire de communication pour récupérer les caractéristiques du protocole de communication permettant d'accéder à un Composant Elémentaire.

Cet attribut est systématiquement disponible sur le nœud racine.

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

Il est accessible en lecture ou en écriture.

#### Java

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

#### COM

- Type `String`
- Code interne `locationsFile`
- get/set C++ `get non disponible / public void setLocationFile(LPCTSTR f)`

## Choix de l'adaptateur de communication

### Description

Cet attribut permet de définir le mode de communication utilisé en indiquant le nom de la classe (ou dll en COM) **ServerAdapter** choisi (com.ibm.vap.middleware.MiddlewareAdapter, com.ibm.vap.gateway.GatewayAdapter pour Java ou MwAdapter, GwAdapter pour COM) ou en indiquant 'Direct' pour un accès Middleware et 'Gateway' pour un accès via l'adaptateur Gateway.

En Java, il est également possible de passer directement une instance **ServerAdapter**.

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)**

#### COM

- Type **String**
- Code interne **serverAdapterName**
- get/set **public BSTR getServerAdapterName ()  
public void setServerAdapterName(LPCTSTR className)**

## 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 (**Direct** ou **Gateway**).

Le tableau suivant donne la liste exhaustive des paramètres acceptés pour les deux modes de communication sus-cités.

Note : pour le détail de ces paramètres, reportez-vous à la fin de cette section.

Paramètres communs aux deux modes de communication	Paramètres Middleware	Paramètres Gateway
<code>folder and location</code>	<code>locationsFile</code>	<code>host</code>
<code>userId</code>	<code>traceFile</code>	<code>port</code>
<code>password</code>	<code>traceLevel</code>	
<code>connectionCleaningInterval</code>	<code>nbMaxConnection</code>	
<code>hostEncoding</code>	<code>connectionTimeout</code>	
<code>clientEncoding</code>	<code>codePageFile</code>	

### 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)
```

### COM

- Type `Long Double String`
- Code interne `setDoubleProperty setIntProperty setStringProperty`
- Nom d'utilisation `setDoubleProperty setIntProperty setStringProperty`
- get/set `get non disponible /`

```
public void setIntProperty(LPCTSTR attribut_name, Long
value)
public void setDoubleProperty(LPCTSTR attribut_name,
double value)
public void setStringProperty(LPCTSTR attribut_name,
LPCTSTR value)
```



Vous trouverez ci-dessous le détail des paramètres acceptés pour les modes de communication 'Direct' et 'Gateway'.

### **ClientEncoding**

Nom et code du jeu de caractères utilisé par le programme client. Si cette propriété est définie, les caractères Unicode à envoyer au serveur, seront d'abord convertis dans ce code page "client" puis convertis en code page serveur (selon l'encodage serveur) et finalement envoyés.

Il n'est pas nécessaire de définir la propriété ClientEncoding si celle-ci n'est pas compatible avec la HostEncoding (i.e. elle contient les mêmes caractères mais peut-être pas avec les mêmes codes).

Dès que l'encodage utilisé par le programme client N'EST PAS compatible avec l'encodage serveur (i.e. ne contient pas le même jeu de caractères), la propriété encodage client doit être définie pour éviter la perte de caractères. Valeur par défaut : nulle, ce qui signifie que les caractères seront directement convertis de Unicode en page de code serveur.

### **CodePageFile**

Nom du fichier contenant les tables de conversion des codes page. Facultatif.

Défaut : fichier CharConv.txt que l'on trouve dans le fichier de travail courant s'il existe. Si aucun nom de fichier n'est donné et que le fichier de conversion n'est pas trouvé, aucune conversion de caractères ne sera faite.

### **ConnectionCleaningInterval**

Temps en millisecondes entre deux nettoyages de connexions serveur inactives.

Pour des raisons de performance, la couche middleware gère un groupe de connexions serveur associées. Une connexion inactive est une connexion dans le groupe qui n'a pas été utilisée depuis le dernier nettoyage. Affectez une petite valeur à ce paramètre (par exemple 1 seconde, i.e. 1000 ms), si vous ne souhaitez pas conserver de connexions au serveur qui sont inutilisées (afin de limiter l'utilisation des ressources).

Affectez une valeur importante à ce paramètre (par exemple 60 secondes), si vous souhaitez obtenir de meilleures performances (réduire le nombre de connexions/déconnexions/re-connexions).

Valeur par défaut : 60 secondes (i.e. 60000).

### **ConnectionTimeout**

Indique le temps maximum en millisecondes qu'un flot d'exécution en attente d'une connexion (lorsque "nbMaxConnection" est atteint) va attendre avant de renvoyer une erreur de communication à l'application. Si vous affectez une petite valeur à ce paramètre (i.e. 1 seconde ou moins), l'application sera susceptible de se bloquer. Dès que le nombre maximum de connexions est atteint, les erreurs de communication vont avoir lieu. Si vous affectez une grande valeur à ce paramètre (i.e. l'infini), l'adaptateur ne renverra jamais d'erreurs dues à de longs temps d'attente (ainsi l'application ne détectera pas les longues attentes).

Défaut: infini.

**Folder and location**

Nom du point d'entrée (Localisation) dans le fichier des localisations où l'on recherche les autres paramètres de communication.

Le nom de la localisation détermine la section qui doit être utilisée. Un Dossier donné n'est utilisé que si aucune localisation n'est donnée. Dans ce cas, la localisation active sera la première à contenir une propriété portant le nom FOLDER dont la valeur est égale au Dossier donné. Dans le cas où ni le Dossier et ni la localisation sont spécifiés, c'est la première localisation dans le fichier de localisations qui sera utilisée.

**Host**

Nom ou adresse IP du host où la gateway est installée.

Défaut : 127.0.0.1 pour host local.

**HostEncoding**

Nom ou code de l'encodage caractères utilisé par le serveur.

Les caractères à envoyer seront convertis dans un code page donné avant d'être envoyés.

La valeur du paramètre HostEncoding devra être soit un code page IBM (ex: "37", "297") connu dans le fichier actif de conversion de caractères (voir le paramètre "codePageFile"), soit une valeur code page précédée de "Cp" (ex: "Cp37", "Cp297"), soit un faux nom (nom d'emprunt) défini dans le fichier de conversion des caractères.

Si ce paramètre n'est pas positionné, la valeur du HOST\_ENCODING (anciennement nommé MWCODE) qui se trouve dans la localisation active dans le fichier des localisations sera utilisée.

Si ce paramètre n'est pas positionné et qu'aucun paramètre HOST\_ENCODING (et aucun MWCODE) n'est trouvé dans la localisation active dans le fichier des localisations, les caractères ne seront pas convertis avant d'être envoyés.

**LocationsFile**

Nom du fichier utilisé pour rechercher d'autres paramètres de communication.

Le contenu du fichier de localisations doit être organisé en sections principales (appelées localisations, encadrées par des < >). Chaque section Location contient les attributs de communication pour accéder à un hôte spécifique.

Le nom de fichier par défaut est "vaplocat.ini" du répertoire de travail courant. Si aucun nom de fichier valide n'est donné et que le fichier par défaut n'est pas trouvé, une erreur de communication sera envoyée.

**NbMaxConnection**

Ce paramètre indique le nombre maximum de connexions serveur qui peuvent exister en même temps.

Avant de créer une nouvelle connexion (lorsqu'il n'y a pas de connexion inactive correspondante dans le groupe), l'adaptateur vérifie d'abord que le nombre maximum de connexions ne soit pas atteint avant de créer une nouvelle connexion. S'il est atteint, la dernière connexion inactive utilisée est détruite avant la création de la nouvelle. Si toutes les connexions sont utilisées (i.e. il n'y a aucune connexion à détruire), le flot d'exécution courant est bloqué jusqu'à ce qu'une connexion soit libérée. Par exemple, si vous positionnez ce paramètre à 1, seule une connexion sera créée. Lorsque deux flots ont besoin de communiquer « en même temps » via la même connexion (i.e. mêmes paramètres de connexion), le second flot d'exécution attendra son tour pour utiliser la connexion.

Lorsque deux flots ont besoin de communiquer « en même temps » via deux connexions différentes (i.e. paramètres de connexion différents), le second flot d'exécution attendra que le premier flot ait fini d'utiliser sa connexion qui sera ensuite détruite avant la création de la connexion nécessitée par le deuxième flot d'exécution. Ce paramètre a un impact important sur la performance.

Défaut : infini (aucun maximum, jusqu'à ce que la communication API associée abandonne).

### **Password**

Mot de passe utilisé pour communiquer via le middleware.

Si ce paramètre n'est pas spécifié, aucun mot de passe ne sera transmis à la couche middleware.

### **Port**

Valeur du port IP où la gateway réceptionne les requêtes clientes.

Défaut : 5647, valeur par défaut du port utilisé par la gateway au démarrage.

### **TraceFile**

Nom du fichier qui réceptionne la trace d'exécution du middleware.

Le fichier par défaut est un nom de fichier créé automatiquement (avec timestamp) dans le sous-répertoire VapTrace du répertoire de travail courant.

### **TraceLevel**

Niveau de détail de la trace d'exécution du middleware :

0 : pas de trace

1 : trace des erreurs uniquement

2 : trace standard non détaillée

3 : trace pour informations

4 et + : trace de debug.

Défaut : 1.

### **UserId**

Code utilisateur utilisé pour communiquer via le middleware.

Si ce paramètre n'est pas spécifié, aucun code utilisateur ni mot de passe ne sera transmis à la couche middleware .

---

## Gestion des conversations asynchrones

### 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)**

#### COM

- Type **Boolean**
- Code interne **asynchronous**
- get/set C++ **public BOOL getAsynchronous ()  
public void setAsynchronous (BOOL a)**

### 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**

#### COM

- Type **ServerActionContext**
- Code interne **lastReplyContext**
- get/set C++ **public LPDISPATCH getLastReplyContext ()  
set non disponible**

## Nombre maximum de réponses en attente

### Description

Cet attribut contient le nombre maximum de requêtes en attente de réponses 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()`  
`public void setMaximumReplyCount(int maximumReplyCount)`

#### COM

- Type `int`
- Code interne `maximumReplyCount`
- get/set C++ `public short getMaximumReplyCount()` / set non disponible

## 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 réponses ou lorsque les 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

#### COM

- Type `int`
- Code interne `pendingReplyCount`
- get/set C++ `public short getPendingReplyCount()` / set non disponible

---

## Temps de conversation

### 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 zéro.

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

#### COM

- Type `long`
- Code interne `communicationResponseTime`
- get/set C++ `public long getCommunicationResponseTime()`  
set non disponible

### 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 zéro.

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

#### COM

- Type `long`
- Code interne `serverResponseTime`
- get/set C++ `public long getServerResponseTime()`  
set non disponible

---

## Gestion des sous-schémas

### 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 Élémentaires 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

#### COM

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

- Code interne **getSubSchemas**
- Signature **public VapCollection getSubSchemas ()**  
Disponible également avec une API de parcours de collection
- Nb d'éléments **public long getSubSchemasCount ()**
- Élément **public LPDISPATCH getSubSchemasElementAt (long i)**

## 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 Élémentaires 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)**

### COM

- Type **VapSubSchema**
- Code interne **subSchema**
- get/set C++ **LPDISPATCH getSubSchema() / void setSubSchema(LPDISPATCH s)**



---

## Gestion des requêtes externes

### Accès et affectation d'une requête

#### Description

Cet attribut retourne ou positionne la requête en cours pour la Proxy.

La méthode d'affectation permet de faire participer la Proxy à un contexte de stockage des actions démarrées par une autre instance de Proxy.

#### Java

- Type `MainRequest`
- Code interne `request`
- Nom d'utilisation `MainRequest`
- get/set `public MainRequest getRequest() / public void setRequest(MainRequest request)`

#### COM

- Type `VapRequest`
- Code interne `Request`
- get/set C++ `LPDISPATCH getRequest() / void setRequest(LPDISPATCH)`

---

## Utilisation d'une JTable

### 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 nœuds 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)**

#### COM

Non disponible

### 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 modifiés 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)**

#### COM

Non disponible

## 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)**

#### COM

Non disponible

## 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 nœuds 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)**

#### COM

Non disponible



---

## Chapitre 3. Actions

Une action correspond à un traitement que la Proxy 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 des Vues Logiques.

Les actions d'une Proxy 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.
- 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 ou des traitements distants. Ce sont les traitements standards de sélections et de mises à jour et les traitements utilisateur définis sur les Composants Elémentaires associés aux Vues Logiques.

**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.

---

## Actions exécutées localement

### Mises à jour

#### 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 Élémentaire 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`

#### COM

- Signature C++ `public void createInstance()`
- Code interne `createInstance()`

## 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 Élémentaire permet les mises à jour sur la Vue Logique.

## Java

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

## COM

- Signature C++ `public void modifyInstance()`
- Code interne `modifyInstance()`

## 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 Élémentaire permet les mises à jour sur la Vue Logique.

## Java

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

## COM

- Signature C++ `public void deleteInstance()`
- Code interne `deleteInstance()`

## Annulation des mises à jour

### Annulation des mises à jour d'un Dossier

#### Comportement

Cette action permet d'annuler toutes les mises à jour locales effectuées sur une instance du 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 listes 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 Élémentaires du Dossier gère une Vue Logique en mise à jour.

## Java

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

## COM

- Signature C++ `public void undoLocalFolderUpdates(LPDISPATCH d)`
- Code interne `undoLocalFolderUpdates({Nom de la Classe DataUpdate générée})`



## Annulation des mises à jour de toutes les instances du Dossier

### Comportement

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

Après l'action :

- les images initiales des instances du Dossier et de toutes leurs instances dépendantes sont restaurées dans le cache local, dans les attributs de présentation et dans les containers de listes d'instances,
- le compteur du nombre total de services de mise à jour est réinitialisé,
- les compteurs du nombre de services de mises à jour associé à chaque nœud sont réinitialisés,
- le compteur du nombre total d'instances locales est recalculé,
- l'attribut de présentation des instances modifiées du Dossier 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 Élémentaires du Dossier gère une Vue Logique en mise à jour.

#### Java

- Signature `public void undoAllLocalFolderUpdates ()`
- Nom d'utilisation `Undo All Local Folder Updates`

#### COM

- Signature C++ `public void undoAllLocalFolderUpdates ()`
- Code interne `undoAllLocalFolderUpdates ()`

## Annulation des mises à jour d'une instance de nœud

### 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 (si l'instance est au statut #Deleted ou #Created) est restaurée dans le cache local et dans les attributs de présentation 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 mise à jour et toutes les instances dépendantes sont supprimées des attributs de présentation d'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 Élémentaire associé au nœud comporte un service de mise à jour.

### Java

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

### COM

- Signature C++ `public void undoLocalUpdate(LPDISPATCH d)`
- Code interne `undoLocalUpdate({Nom de Classe DataDescriptionUpdate générée} d)`

## Annulation des mises à jour de toutes les instances d'un nœud

### Comportement

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

Après l'action :

- les images initiales des instances du nœud 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 réinitialisé,
- le compteur du nombre total d'instances locales est recalculé,
- les instances mises à jour et toutes leurs instances dépendantes sont supprimées des attributs de présentation d'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 Élémentaire associé au nœud comporte un service de mise à jour.

#### Java

- Signature `public void undoAllLocalUpdate()`
- Nom d'utilisation `Undo All Local Update`

#### COM

- Signature C++ `public void undoAllLocalUpdate()`
- Code interne `undoAllLocalUpdate()`

## Gestion des Services Utilisateur

### 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 Élémentaire associé au nœud gère au moins un Service Utilisateur.

#### Java

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

#### COM

- Signature C++ `public void createUserInstance()`
- Code interne `createUserInstance()`

## 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 Élémentaire associé au nœud gère au moins un Service Utilisateur.

#### Java

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

#### COM

- Signature C++ `public void modifyUserInstance ()`
- Code interne `modifyUserInstance ()`

## 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 Élémentaire associé au nœud gère au moins un Service Utilisateur.

### Java

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

### COM

- Signature C++ `public void deleteUserInstance()`
- Code interne `deleteUserInstance()`

## Navigation locale dans les Dossiers

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

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

- si le nœud dépendant a une cardinalité maximale de n, son attribut **rows** 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 **detail** est aussi alimenté par cette instance.,
- si le nœud a une cardinalité maximale de 1, son attribut **detail** 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, ses attributs de présentation d'instances sont vides.

### Sélection courante d'une instance dans un Dossier

#### Comportement

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

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 **detail** contient l'instance à affecter,
- les attributs **detail** et **rows** des nœuds dépendants sont alimentés suivant la règle globale d'alimentation,
- l'identifiant de verrouillage du Dossier est alimenté si l'instance traitée est une instance du 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 DataDescription générée} d) throws LocalException`
- Nom d'utilisation `Get Detail From Data Description`

#### COM

- Signature C++ `public void getDetailFromData(LPDISPATCH d)`
- Code interne `getDetailFromData({Nom de classe DataDescription générée})`

## Sélection d'une instance à partir d'un index

### Comportement

Cette méthode permet de remettre en place la sélection courante selon l'index d'une instance de Vue Logique contenu dans la collection "rows".

#### Java

- Signature `public void getDetailFromRowIndex(int index) throws LocalException`
- Nom d'utilisation `Get Detail From Rows Index`

#### COM

- Signature C++ `public void getDetailFromRowIndex (short index)`
- Code interne `getDetailFromRowIndex (Integer)`

## Sélection d'une instance associée à un Service Utilisateur

### Comportement

Cette action permet d'affecter à l'attribut **UserDetail** (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 **UserRows** (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 Élémentaire associé au nœud a au moins un Service Utilisateur.

#### Java

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

#### COM

- Signature C++ `public void getUserDetailFromData (LPDISPATCH d)`
- Code interne `getUserDetailFromData({Nom de classe DataDescription générée} d)`



## Remise en place de la sélection courante

### Comportement

Cette action permet de remettre en place la sélection courante selon une instance de Vue Logique.

Remarque : L'instance de Vue Logique n'est pas forcément obtenue à partir de la collection **rows**, elle a pu être créée seulement pour les besoins du développeur.

Exemple : après une sélection de 300 instances de Vue Logique "Client" puis en créant une instance de la Vue Logique "Client", en lui positionnant le numéro de client à 56 et en utilisant la méthode "restoreSelection", la sélection courante de la Proxy sera alimentée avec le client 56 ramené préalablement et la hiérarchie des dépendants sera mise en place avec le client 56 comme racine.

#### Java

- Signature `public void restoreSelection ({Nom de classe générée}Data d)`
- Nom d'utilisation `restoreSelectionFromData ({Nom de classe générée}Data d)`

#### COM

Non disponible

## Initialisations diverses

### 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 **detail** du nœud est réinitialisé,
- l'attribut **rows** du nœud est réinitialisé,
- les attributs **detail** des nœuds dépendants sont réinitialisés,
- les attributs **rows** des nœuds dépendants sont réinitialisés.

Cette action est systématiquement disponible pour tous les nœuds.

#### Java

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

#### COM

- Signature C++ `public void resetCollection()`
- Code interne `resetCollection()`

## Initialisation des méthodes d'extraction

### Comportement

Cette action permet d'initialiser l'attribut **méthode d'extraction à exécuter** du nœud 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 disponible pour les nœuds sur lesquels au moins une méthode d'extraction a été définie dans le composant élémentaire.

#### Java

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

#### COM

- Signature C++ `public void resetExtractMethodCodes ()`
- Code interne `resetExtractMethodCodes ()`

## 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 Élémentaire associé au nœud a au moins un Service Utilisateur.

#### Java

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

#### COM

- Signature C++ `public void resetUserServiceCodes ()`
- Code interne `resetUserServiceCodes ()`

## 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 Élémentaire associé au nœud a au moins un Service Utilisateur.

#### Java

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

#### COM

- Signature C++ `public void resetUserRows ()`
- Code interne `resetUserRows ()`

## 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 Élémentaire associé au nœud utilise la Vue Logique en mise à jour.

#### Java

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

#### COM

- Signature C++ `public void resetAllRefreshOption ()`
- Code interne `resetAllRefreshOption ()`

## 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 racines et dépendants.

#### Java

- Signature `public void resetSelectionCriteria()`
- Nom d'utilisation `Reset Selection Criteria`

#### COM

- Signature C++ `public void resetSelectionCriteria()`
- Code interne `resetSelectionCriteria()`

## 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. Cette instance n'a pas le statut « créée localement ».

Cette action est valide si l'instance n'existe pas en local, quel que 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 nœud 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 `Initialize Instance`

#### COM

- Signature C++ `public void initializeInstance()`
- Code interne `initializeInstance()`

## Gestion des instances référencées

### Affectation d'une instance référencée

#### Comportement

Cette action permet d'affecter les Rubriques de type identifiant de l'instance du nœud référence passée en paramètre 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 références

#### Java

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

#### COM

- Signature C++ `public void transferReferenceFromSelectedRow(LPDISPATCH d)`
- Code interne `transferReferenceFromSelectedRow({Nom de classe DataDescription générée} d)`

## Récupération des contextes de génération des Proxies

### 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 Élémentaire),
- ◆ 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`

#### COM

Disponible avec une API de parcours de collection.

- Nb d'éléments `public Long getFolderConstantsCount()`
- Élément `public String getFolderConstantsElementAt(Long i)`

## Contexte de génération d'un nœud

### Comportement

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

- ◆ nom externe du Composant Élémentaire,
- ◆ code VisualAge Pacbase du Composant Élémentaire,
- ◆ 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 Élémentaire.

Cette action est systématiquement disponible pour tous les nœuds (racines, dépendants et références).

#### Java

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

#### COM

- Nb d'éléments `public Long getNodeConstantsCount()`
  - Élément `public String getNodeConstantsElementAt(Long i)`
- Disponible avec une API de parcours de collection.

## Gestion des sous-schémas

### 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 Élémentaires 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`

#### COM

- Signature C++ `public resetSubSchema()`
- Code interne `resetSubSchema()`

---

## 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.

### Sélection sur un nœud

#### 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 racines et références.

#### Java

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

#### COM

- Signature C++ `public void selectInstances ()`
- Code interne `selectInstances ()`

### 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 locale 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 listes 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,
- 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`

#### COM

- Signature C++ `public void readInstance()`  
`public void readInstanceAndLock()`
- Code interne `readInstance()`  
`readInstanceAndLock()`

### Lecture d'instances à partir d'identifiants

Cette action permet de définir une collection d'instances de Vue Logique associée au nœud et de récupérer les instances dont les clés sont passées en paramètre.

La collection de clés passée en paramètre peut-être constituée d'instances de classes `SelectionCriteria` ou de classes `DataDescription`.

Si cette action est valide :

- l'attribut `rows` est renseigné selon le mode de gestion de collection,
- le compteur d'instances locales du Dossier est initialisé,
- le libellé et la clé du message de retour de sélection sont initialisés si la dernière instance a été récupérée,
- les attributs `detail` et `rows` des nœuds dépendants sont initialisés à vide,
- 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 systématiquement disponible sur tous les nœuds.

## Java

- Signature `public void readInstances(Enumeration keys) throws ServerException, LocalException, CommunicationError, SystemError`
- Nom d'utilisation `Read Instances`

## COM

- Signature C++ `public void readInstances(LPDISPATCH keys)`
- Code interne `readInstances (VapCollection)`

## Sélection simultanée sur plusieurs nœuds avec ou sans verrouillage

### 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 a 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 nœud,
- 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 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 tous les nœuds racines 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`

#### COM

- Signature C++ `public void readWithFirstChildren()`  
`public void readWithFirstChildrenAndLock()`
- Code interne `readWithFirstChildren()`  
`readWithFirstChildrenAndLock()`

### 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 nœud,
- 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
```

## COM

- Signature C++ 

```
public void readWithAllChildren()
public void readWithAllChildrenAndLock()
```

- Code interne `readWithAllChildren()`  
`readWithAllChildrenAndLock()`

## 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 racines et dépendants.

### Java

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

### COM

- Signature C++ `public void readFirstChildrenFromDetail()`
- Code interne `readFirstChildrenFromDetail()`

## 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'attribut **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 les nœuds racines.

### Java

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

## COM

- Signature C++ `public void readAllChildrenFromDetail()`
- Code interne `readAllChildrenFromDetail()`

### 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`

## COM

- Signature C++ `public void readWithFirstChildrenFrom(LPDISPATCH d)`
- Code interne `readWithFirstChildrenFrom({Nom de classe DataDescription générée} d)`

### 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 DataDescription générée} d) throws LocalException, ServerException, CommunicationError, SystemError`
- Nom d'utilisation `Read All Children From {Nom de classe générée}Data`

## COM

- Signature C++ `public void readWithAllChildrenFrom(LPDISPATCH d)`
- Code interne `readWithAllChildrenFrom({Nom de classe DataDescription générée} d)`

## Gestion de la pagination

### 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`

#### COM

- Signature C++ `public void readNextPage()`
- Code interne `readNextPage()`

## 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'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 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`

#### COM

- Signature C++ `public void readPreviousPage()`
- Code interne `readPreviousPage()`

## Emission des mises à jour

### 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 Élémentaires 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`

#### COM

- Signature C++ `public void updateFolder()`
- Code interne `updateFolder()`

## Gestion du verrouillage logique

### 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`

#### COM

- Signature C++ `public void lock()`
- Code interne `lock()`

## 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`

### COM

- Signature C++ `public void unlock()`
- Code interne `unlock()`

## Gestion des instances dépendantes

### 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`

#### COM

- Signature C++ `public void checkExistenceOfDependencies()`
- Code interne `checkExistenceOfDependencies()`

## Gestion des Services Utilisateur

### 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 Élémentaire 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`

#### COM

- Signature C++ `public void executeUserService()`
- Code interne `executeUserService()`

## Gestion des conversations asynchrones

### 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.ServerActionContext aContext)  
throws LocalException, ServerException, CommunicationError,  
SystemError
```
- Nom d'utilisation `Get Reply`

#### COM

- Signature C++ 

```
public BOOL getReply(LPDISPATCH i)
```
- Code interne 

```
Boolean getReply(<nomDossier>ServerAction)
```



## 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 renvoie **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`

### COM

- Signature C++ `public BOOL isReplyValid(LPDISPATCH i)`
- Code interne `Boolean isReplyValid(<nomDossier>ServerAction)`

## Gestion des sous-schémas

### Comportement

Cette action permet de récupérer, par appel du Composant Élémentaire 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 Élémentaires 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`

### COM

- Signature C++ `public completeInstance()`
- Code interne `completeInstance()`

## Test de communication avec le serveur

### Comportement

Cette action permet de faire un test de communication vers le moniteur de communication et ainsi de valider les paramètres de communication sans avoir à faire un accès aux serveurs élémentaires.

#### Java

- Signature `public void ping() throws CommunicationError`
- Nom d'utilisation `Ping`

#### COM

- Signature C++ `public void ping()`
- Code interne `ping()`

---

## Externalisation de la gestion des requêtes

Le principe d'externalisation de la gestion des requêtes est de créer un contexte de stockage des actions à exécuter sur le serveur et donc de permettre l'envoi de services différents dans une même requête serveur.

Ce contexte se définit au travers d'un objet de type "MainRequest" qui permet de "stocker" et de "poster" des services différents, provenant de plusieurs Proxies ou non, afin qu'ils soient envoyés dans le même échange serveur. Cela permet notamment de partager le même contexte transactionnel dans le cas d'envoi de mises à jour sur plusieurs Dossiers.

### Création d'une requête

#### Comportement

Cette action permet de démarrer un contexte de stockage des actions à exécuter sur le serveur en créant une instance de MainRequest pour la Proxy. Cette action valorise l'attribut "request" pour la Proxy. L'ensemble des actions devant être exécutées sur le serveur seront d'abord "stockées" localement dans l'objet requête.

#### Java

- Signature `public void createRequest()`
- Nom d'utilisation `Create Request`

#### COM

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

## Exécution sur le serveur des actions de la requête

### Comportement

Cette action permet d'exécuter sur le serveur l'ensemble des actions qui ont été "stockées" dans la requête.

Cette action est disponible sur l'objet MainRequest

#### Java

- Signature `public void sendRequest() throws ServerException, LocalException, CommunicationError, SystemError`
- Nom d'utilisation `Send Request`

#### COM

- Signature C++ `public void sendRequest()`
- Code interne `sendRequest()`

## Abandon des actions de la requête

### Comportement

Cette action permet d'abandonner l'ensemble des actions qui ont été "stockées" dans la requête.

Cette action est disponible sur l'objet MainRequest.

#### Java

- Signature `public void cancel()`
- Nom d'utilisation `Cancel`

#### COM

- Signature C++ `public void cancel()`
- Code interne `cancel()`

---

## Manipulation de flots XML

Le flot XML permet de représenter :

- la hiérarchie d'informations pour un nœud du dossier (Detail),
- la collection d'instances pour un nœud du dossier (Rows).

Dans les deux cas, les flots XML correspondront à la structure hiérarchique du dossier, à partir de la racine du dossier.

Pour chaque instance représentée, le flot doit décrire :

- l'état de l'instance dans le cache de la proxy,
- la valeur des champs de l'instance, (tous les champs peuvent ne pas être valorisés puisque l'on tient compte de la notion de sous-schéma exprimée lors des opérations de sélection des instances d'une proxy).

## Etat d'une instance de vue logique

Cette notion s'apparente à la notion d'« attribut », défini pour un **complexType** (le nœud) dans un schéma XML. Cet attribut a pour valeurs permises : 'read', 'created', 'modified', 'deleted', ou 'unknown' (cas de récupération d'une instance de détail invalide/inconnue).

Afin d'assurer l'unicité du nom de cet attribut et éviter les conflits avec le nom des champs d'une vue logique, ce nom est exprimé sur plus de 6 caractères : **dataStatus**.

## Obtention d'un flot XML

### Comportement

Cette action permet d'obtenir :

- les instances d'un nœud avec hiérarchie ou non (rows),
- une instance de nœud avec hiérarchie ou non (detail).

Les paramètres passés permettent :

- d'identifier le flot XML,
- d'identifier le nœud dont on veut récupérer l'instance,
- de préciser si l'on veut récupérer des rows (**true**),
- d'indiquer si l'on veut récupérer l'instance de nœud et de sa hiérarchie(**true**).

Cette action est invalide si le contexte de navigation du cache est incohérent.

Si cette action est valide :

- si la demande porte sur une hiérarchie, sauvegarde du contexte de navigation du cache,
- construction du flot XML par parcours de la hiérarchie dans le cache,
- si la demande porte sur une hiérarchie, restauration du contexte de navigation du cache.

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.

### Java

• Signature `public void getXML(java.io.OutputStream outputStream, ProxyLv proxy, Boolean rows, Boolean hierarchicalStructure) throws XMLWrapperException`

### COM

• Signature C++ `public void getXML(utfostream out, ProxyLv proxy, bool rows, bool hierarchicalStructure)`

## Mise à jour à partir d'un flot XML

### Comportement

Dans ce cas, quel que soit le flot XML, celui-ci doit être valide :

- rows avec ou sans hiérarchie,
- detail avec ou sans hiérarchie.

L'état associé à chaque instance de nœud dans le flot XML permet de guider la mise à jour du cache. Cette opération s'effectue en fonction de l'état initial de l'instance dans le cache.

Etat initial de l'instance dans le cache	Etat de l'instance dans le flot XML	Action locale sur la proxy
created	created	modify instance si maj
read	created	exception
modified	created	exception
unknown	created	create instance
deleted	created	create instance
created	read	exception
read	read	rien
modified	read	exception
unknown	read	initialize instance
deleted	read	exception
created	modified	modify instance
read	modified	modify instance
modified	modified	modify instance
unknown	modified	initialize + modify instance
deleted	modified	exception
created	deleted	delete instance
read	deleted	delete instance
modified	deleted	delete instance
unknown	deleted	initialize + delete instance
deleted	deleted	rien

**Remarque :** La gestion du lock reste à la charge de l'utilisateur et doit être effectuée le cas échéant avant mise à jour du cache à partir d'un flot XML.

Cette action est invalide si le contexte de navigation du cache est incohérent.

Si cette action est valide :

- parsing de type SAX du flot XML,
- pour chaque instance de nœud, identification de l'état associé : comparaison avec l'état de cette instance dans le cache avant mise à jour.

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.

## Java

- Signature `public void updateFromXML(java.io.InputStream xmlStream) throws XMLWrapperException`

## COM

- Signature C++ `public void updateFromXML(utfostream xmlStream)`

---

## Gestion du contexte d'une proxy

### Initialisation du contexte d'une proxy

#### Comportement

Cette action permet de mémoriser le contexte actuel et de restituer une instance de l'objet initialisé. Les attributs mémorisés dans ce contexte sont : les attributs de communication, les attributs d'extraction et de pagination, les attributs de mises à jour locales, les attributs des services utilisateurs, les attributs du buffer utilisateur et ceux de l'instance courante.

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 le nœud racine.

## Java

- Signature `public com.ibm.vap.generic.ProxyContext getProxycontext() throws LocalException`
- Nom d'utilisation `getProxyContext`

## COM

- Signature C++ `public VapTools.VapProxyContext getProxycontext()`
- Code interne `getProxyContext`

### Initialisation du cache local

#### Comportement

Cette action permet de repositionner les attributs dans le cache local de la proxy et de ses nœuds dépendants à partir de l'instance ProxyContext passé en paramètre et ainsi de permettre la reprise des lectures et des mises à jour.

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 le nœud racine.

## Java

- Signature `public void  
initFromProxycontext (com.ibm.vap.generic.ProxyContext  
context)  
throws LocalException`
- Nom d'utilisation `initFromProxycontext`

## COM

- Signature C++ `public void initFromProxycontext (VapTools.VapProxyContext  
context)`
- Code interne `initFromProxycontext`

## Récupération de la proxy

### Comportement

Cette méthode permet de récupérer le nom de la proxy avec laquelle l'objet ProxyContext a été initialisée afin de pouvoir la ré-instancier.

## Java

- Signature `public String getProxyClassName ()`
- Nom d'utilisation `getProxyClassName`

## COM

- Signature C++ `public BSTR getProxyClassName ()`
- Code interne `getProxyClassName`






---

## Chapitre 4. Événements

---

### Gestion de la pagination

-  Pour la cible COM, tous les événements décrits dans ce chapitre sont stockés dans une pile (consulter le paragraphe concernant *Gestion des événements*).

#### 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. Cet événement est disponible pour les nœuds racines, références et pour les nœuds dépendants de cardinalité maximale n.

##### Java

- Code `noPageAfter`

##### COM

- Code `NO_PAGE_AFTER`

#### 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'instances ou de pagination renvoie une page contenant la première instance de la collection. Cet événement est disponible pour les nœuds racines et références lorsque le mode de pagination est de type `non-extend` avec un mode de gestion de collections automatique.

##### Java

- Code `noPageBefore`

##### COM

- Code `NO_PAGE_BEFORE`

#### 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'instances ou de pagination renvoie une page ne contenant pas la dernière instance de la collection. Cet événement est disponible pour les nœuds racines, références et pour les nœuds dépendants de cardinalité maximale n.

##### Java

- Code `pageAfter`

##### COM

- Code `PAGE_AFTER`

## 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'instances ou de pagination renvoie une page ne contenant pas la première instance de la collection.

Cet événement est disponible pour les nœuds racines et références lorsque le mode de pagination est de type **non-extend** et en mode de gestion de collection automatique.

#### Java

- Code `pageBefore`

#### COM

- Code `PAGE_BEFORE`

---

## Gestion des lectures unitaires

### 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`

#### COM

- Code `NOT_FOUND`

---

## Gestion des sélections simultanées

### Signal de non participation à une lecture simultanée

#### Règles d'émission

Ce signal est émis par un nœud suite à une action de lecture/écriture à laquelle le nœud n'a pas participé.

Cette action est systématiquement disponible pour tous les nœuds.

#### Java

- Code `notRead`

#### COM

- Code `NOT_READ`

---

## Gestion du verrouillage logique

### Signal de verrouillage logique effectué

#### Règles d'émission

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

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`

#### COM

- Code `LOCK_SUCCESSFUL`

### Signal de verrouillage logique infructueux

#### Règles d'émission

Ce signal est émis par un nœud racine après l'échec d'une action de verrouillage logique d'une instance. Cette instance étant 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`

#### COM

- Code `LOCK_FAILED`

---

## Gestion des instances dépendantes

### 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`

#### COM

- Code `DEPENDENT_INSTANCES`

### Signal d'absence d'instances dépendantes

#### 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 racines et dépendants.

#### Java

- Code `noDependentInstances`

#### COM

- Code `NO_DEPENDENT_INSTANCES`

---

## Chapitre 5. Interface publique de manipulation des Rubriques

---

### 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)`

#### COM

- Type `Dépend du type de la Rubrique`
- Code interne `<CodeRubrique>`
- get/set C++ `public [Type] get<CodeRubrique> ()`  
`public void set<CodeRubrique> (Type t)`

---

### 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`.

#### Java

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

#### COM

- Nb d'éléments `Disponible avec une API de parcours de collection`  
`public long <CodeRubrique>ValidValuesCount ()`
- Élément `public Type <CodeRubrique>ValidValuesAt (long i)`

---

## 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`.

#### Java

- Type `String [ ]`
- Code interne `<CodeRubrique>Labels`
- Nom d'utilisation `<CodeRubrique> Labels`
- get/set `public String[ ] get<CodeRubrique>Labels ( )`  
set non disponible.  
Ces méthodes peuvent accepter la locale cible, nouveau paramètre qui correspond à la langue pour laquelle on veut retrouver le libellé.

#### COM

- Nb d'éléments Disponible avec une API de parcours de collection  
`public long <CodeRubrique>ValidLabelsCount ( )`
- Élément `public Type <CodeRubrique>ValidLabelsAt (long i)`

---

## 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`.

#### Java

- 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.

---

## Accès aux caractéristiques d'une Rubrique

### Description

La classe **DataGroup** offre des méthodes permettant de récupérer les caractéristiques de saisie des champs de Rubriques composant une Vue Logique.

- `findDataFieldFormat` : renvoie le format Va Pac de la Rubrique. (Ne renvoie rien dans le cas d'une Rubrique alphanumérique),

- `findDataFieldMaxLength` : renvoie la longueur maximale autorisée pour la valeur d'une Rubrique.

Ces actions sont systématiquement disponibles pour toutes les Rubriques qui sont définies dans la classe **DataGroup**.

#### Java

- Signature `public java.text.Format findDataFieldFormat(String code)`
- Nom d'utilisation `findDataFieldFormat`

#### COM

Non disponible.

#### Java

- Signature `public int findDataFieldMaxLength (String code)`
- Nom d'utilisation `findDataFieldMaxLength`

#### COM

Non disponible.

---

## Initialisation de l'ensemble des valeurs des Rubriques composant une Vue Logique

### Comportement

Cette méthode permet d'initialiser les valeurs d'une instance de **DataGroup** avec les valeurs d'une autre instance de **DataGroup** passée en paramètre.

#### Java

- Signature `public void initializeFrom(DataGroup)`
- Nom d'utilisation `Initialize From`

#### COM

Non disponible.

---

## Gestion de la présence d'une Rubrique

### Description

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

Ces actions sont systématiquement disponibles pour les Rubriques définies dans les classes **DataDescription** et **UserDataDescription**.

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

#### COM

- Signature C++ `public bool is<CodeRubrique>Present ()`  
`public void set<CodeRubrique>Present (bool b)`
- Code interne `Boolean`  
`is<CodeRubrique>Present/setCodeRubriquePresent (Boolean value)`

---

## 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 racines ou dépendants dont le Composant Élémentaire 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`

#### COM

- Signature C++ `public void setCheck(long fieldIndex, BOOL b)`
- Code interne `setCheck(Long, Boolean)`



---

## Gestion d'appartenance à un sous-schéma

### Comportement

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

Cette action est disponible si les Composants Elémentaires 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`

#### COM

- Signature C++ `public BOOL belongsToSubschema(short index)`
- Code interne `Boolean belongsToSubSchema(Integer)`

---

## Gestion d'appartenance à une méthode d'extraction

Cette action permet de savoir si la Rubrique passée en paramètre appartient à la méthode d'extraction également passée en paramètre.

Cette action est disponible sur les classes **SelectionCriteria** dont le nœud dispose d'au moins une méthode d'extraction.

#### Java

- Signature `public boolean dataFieldBelongsToExtractMethod (String FieldCode, String extractMethodCode)`
- Code interne `DataFieldsBelongsToExtractMethod`

#### COM

- Signature C++ `public BOOL dataFieldBelongsToExtractMethod (LPCTSTR FieldCode, LPCTSTR extractMethodCode)`
- Code interne `Boolean dataFieldBelongsToExtractMethod(String, String)`



---

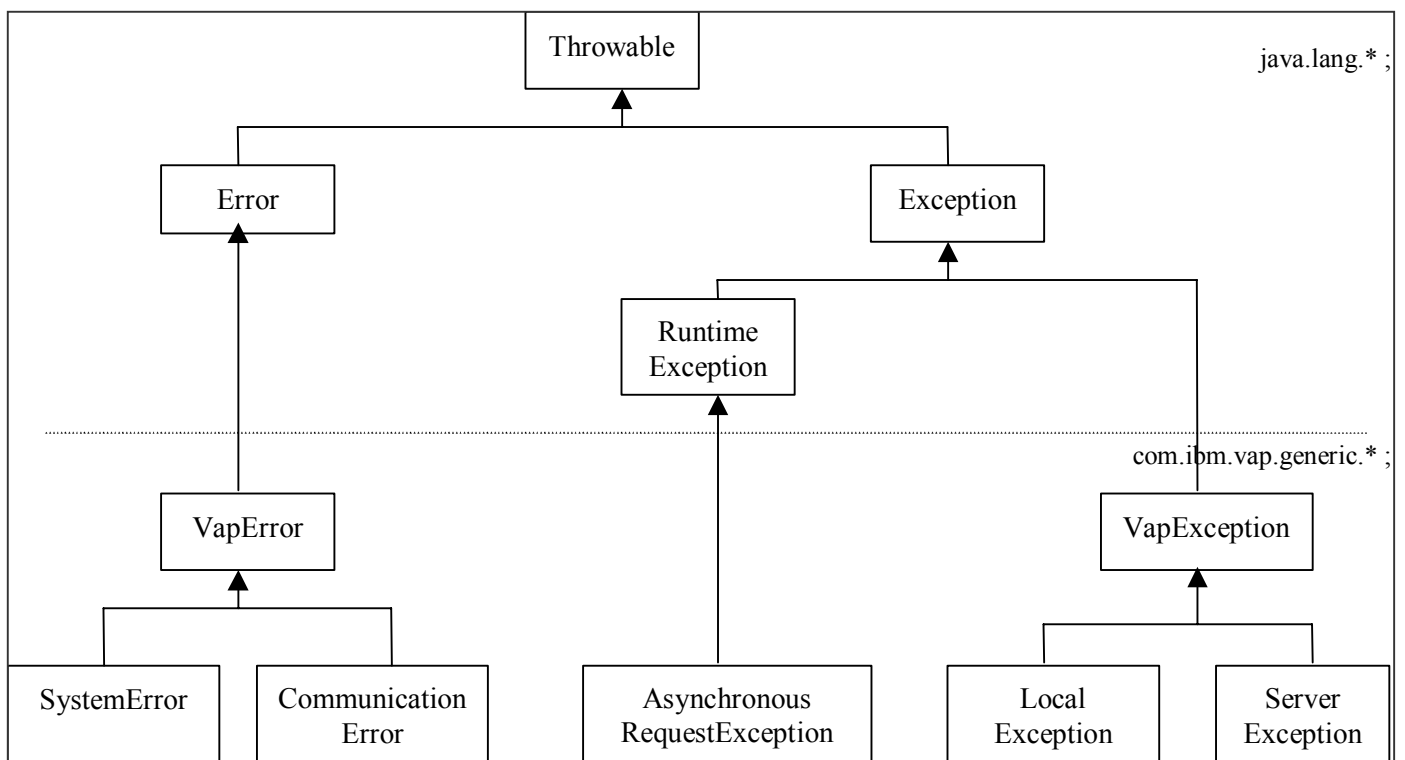
## Chapitre 6. Gestion des Erreurs

---

### 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, propres à ce langage.

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 toutes 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 cela soit optionnel pour ce type de classe dans Java.

**Remarque :** L'utilisation abusive de certaines méthodes de la Proxy peuvent 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).

## Classes relatives à la gestion des erreurs

### 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 composant.

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

### 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 *Applications eBusiness & Pacbench C/S - Clients Graphique*.

### Erreurs locales

#### Nom de l'exception

`com.ibm.vap.generic.LocalException`

**Remarque** Cette exception est provoquée par les erreurs locales. Ces erreurs sont 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 manuel *Applications eBusiness & Pacbench C/S - Clients Graphiques* ainsi que dans la documentation HTML associée aux classes génériques : Package `com.ibm.vap.generic`.

## 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 composant Serveur.

Ces messages d'erreur logiques sont représentés par des instances de `com.ibm.vap.generic.ServerMessage`. La liste de ces messages d'erreur reçus du composant 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()`).

### Message d'erreur reçu du composant Serveur

Les messages d'erreur reçus du composant Serveur sont représentés par des objets de type `com.ibm.vap.generic.ServerMessage`.

Cette interface offre des méthodes permettant notamment de récupérer la clé d'erreur (`String key()`), le libellé du message d'erreur (`String label()`) et le libellé du message d'erreur formaté par le composant Client (`String localLabel()`).

Pour une description du principe de formatage local des libellés d'erreur, reportez-vous au paragraphe *Libellés locaux des messages d'erreur reçus du composant serveur*, page 112.

### Message d'erreur reçu du composant 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.

## 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 formatage 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 `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'erreur 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` formatés localement (`String localLabel()`).

### Libellé des exceptions locales

Les clés permettant de retrouver les libellés des erreurs locales correspondent aux noms des constantes définies sur la classe `com.ibm.vap.generic.LocalException` et qui représentent les différents types d'erreurs locales, préfixés par `LOCAL_`.

**Exemple :** La clé permettant de retrouver le libellé associé à une exception locale de type `INVALID_INSTANCE` sera `LOCAL_INVALID_INSTANCE`.

### Libellés locaux des messages d'erreur reçus du composant serveur

Les messages d'erreur 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 manuel *Applications eBusiness & Pacbench C/S - Clients Graphique*.

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 (différents de blanc) ou bien suivi des caractères compris entre les colonnes 10 et 13.

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 ce n'est pas le cas et que 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 au caractère compris entre les colonnes 22 et 25.

### 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`.

### Exemple de fichier de libellés d'erreur

```
# Ce fichier définit les libellés d'erreur locales et serveurs pour le produit
VisualAgePachbase 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 = Cardinalités non respectées {6} (data id: {3})
LOCAL_INSTANCE_ALREADY_LOCKED = L'instance est déjà verrouillée (data id: {3})
LOCAL_CURRENT_USER_INSTANCE_MISSING = Current user instance missing
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-schéma
(valeur: {5}, data id: {3})
LOCAL_FOLDER_USER_CONTEXT_LENGTH_ERROR = Erreur de longueur sur la Rubrique
{4} du buffer utilisateur du Dossier (valeur: {5})
LOCAL_REFERENCE_USER_CONTEXT_LENGTH_ERROR = Erreur de longueur sur la Rubrique
{4} d'un buffer utilisateur d'un nœud référence (valeur: {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 erroné (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.

```

---

## Gestion des erreurs pour la cible COM

En COM il existe une interface **VAPERROR** qui contient les attributs, actions, événements permettant la consultation de tous les types d'erreurs (locales, de communication et serveur). Cette interface est disponible dans la librairie VapTools livrée avec le générateur. Cette librairie doit être enregistrée sur la machine (regsvr32 VapTools.dll) et référencée dans le langage client avant toute utilisation.

### Méthodes d'accès aux erreurs

#### Description

Cette méthode permet de récupérer une instance de l'objet VapError renvoyée par des actions locales ou serveurs.

Méthode disponible sur chaque nœud de la Proxy.

- Signature C++      **public VapError getErrorsElementAt(int i)**
- Code interne        **getErrorsElementAt(long i)**

#### Description

Cette action contient le nombre d'erreurs renvoyées par une action locale ou serveur.

Méthode disponible sur chaque nœud de la Proxy.



- Signature C++ `public Int getErrorsCount()`
- Code interne `getErrorsCount`

## Attributs de VapError

### Gestion du type de l'erreur

#### Description

Cette action permet de récupérer le type de l'erreur "LOCAL", "SERVER" ou "COMMUNICATION".

- Signature C++ `public BSTR getType()`
- Code interne `String getType()`

### Gestion de l'action provoquant l'erreur

#### Description

Cette action permet de récupérer l'action à l'issue de laquelle a été provoquée l'erreur.

- Signature C++ `public BSTR getAction()`
- Code interne `String getAction()`

### Gestion de la clef d'erreur

#### Description

Cette action permet de récupérer la clé de l'erreur (voir le manuel *Applications eBusiness & Pacbench C/S - Clients Graphiques* pour la liste des erreurs).

Il est accessible en lecture uniquement.

- Signature C++ `public BSTR getType()`
- Code interne `String getKey`

### Gestion du libellé de l'erreur

#### Description

Cette action permet de récupérer le libellé de l'erreur par défaut ou personnalisé (cf. le sous-chapitre *Personnalisation des libellés des erreurs* pour connaître la procédure de personnalisation).

Il est accessible en lecture uniquement.

- Signature C++ `public BSTR getLabel()`
- Code interne `String getLabel()`

### Gestion de la gravité de l'erreur

#### Description

Cette action permet de récupérer la gravité de l'erreur, "EXCEPTION" ou "ERROR"

Il est accessible en lecture uniquement.

- Signature C++ `public BSTR getGravity()`
- Code interne `String getGravity`

## Événements liés aux erreurs

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 Long getServerEventsCount()` ne renvoie pas zéro. Les String récupérés correspondent aux codes des événements décrits.

### Signal de non détection d'erreur

#### Règles d'émission

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

Cet événement est systématiquement disponible.

- Code `NO_ERROR`

### 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.

- Code `LOCAL_ERROR`

### 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.

- Code `SERVER_ERROR`

### 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 Élémentaire associé.

Cet événement est systématiquement disponible.

- Code `SYSTEM_ERROR`

## 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 composant client et un composant serveur.

Cet événement est systématiquement disponible.

- Code **FATAL\_ERROR**

## Personnalisation des libellés des erreurs

La personnalisation des libellés des erreurs associées à la manipulation des Proxies est possible avec les Proxies COM. Les différents libellés sont stockés dans un fichier texte (par défaut VapProxyMsg.txt) se trouvant dans un répertoire accessible par la variable d'environnement Path.

### Règles de nommages des fichiers de libellés d'erreur

Le stockage des libellés d'erreur est fait de la manière suivante:

- Autant de fichier texte VapErrorMsg que de langues ciblées. Le nom des fichiers étant construit comme suit : VapProxyMsg\_langue\_pays.txt.
- On cherche d'abord à charger le fichier correspondant au contexte géographique (ce contexte étant fourni par la **Locale** par défaut), puis si il n'est pas trouvé on tente de charger le fichier avec juste l'extension langue (VapProxyMsg\_langue.txt) si ce chargement reste infructueux on charge alors le fichier par défaut VapProxyMsg.txt.
- Ce fichier est structuré sur le mode relation clé-valeur, où la valeur correspond à un libellé. Les libellés d'erreur stockés dans ce fichier sont des messages associés à des erreurs Locales, Système ou Serveur.

### Syntaxe des fichiers de libellés d'erreur

Pour pouvoir modifier ou créer un fichier de libellés d'erreur, il faut respecter quelques règles syntaxiques. Toute erreur de syntaxe entraîne l'invalidité totale du processus d'externalisation des libellés d'erreur.

- Utiliser des guillemets pour délimiter une relation clé-valeur. Cette relation doit se trouver sur une même ligne.
- Toute ligne ne commençant pas par un guillemet n'est pas interprétée.
- La clé et le libellé doivent être encadrés séparément par des guillemets.
- Pour spécifier un attribut variable dans un libellé, utilisez les accolades { et } pour encadrer le numéro d'attribut.

### Libellé des exceptions locales

Les clés permettant de retrouver les libellés des erreurs locales correspondent aux noms des constantes définies pour les exceptions locales représentant les différents types d'erreurs locales, préfixés par **LOCAL\_**.

**Exemple** : La clé permettant de retrouver le libellé associé à une exception locale de type **INVALID\_INSTANCE** sera **LOCAL\_INVALID\_INSTANCE**.

## Libellés locaux des messages d'erreur reçus du composant serveur

Les messages d'erreur 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 manuel *Applications eBusiness & Pacbench C/S - Clients Graphique*.

Dans le cas d'erreurs 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 (différents de blanc) ou bien suivi des caractères compris entre les colonnes 10 et 13.

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 ce n'est pas le cas et que 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 au caractère compris entre les colonnes 22 et 25.

## Libellés globaux des erreurs serveur et système

Le libellé des exceptions serveur et système sont respectivement accessibles par les clés **VAP\_SERVER\_EXCEPTION** et **VAP\_SYSTEM\_ERROR**.

## Exemple de fichier de libellés d'erreur

```
# 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 missing (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 = Cardinalités 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 absente  
LOCAL\_REFERRING\_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-schéma (valeur: {5}, data id: {3})  
LOCAL\_FOLDER\_USER\_CONTEXT\_LENGTH\_ERROR = Erreur de longueur sur une rubrique du buffer utilisateur du dossier: {4} (value: {5})  
LOCAL\_REFERENCE\_USER\_CONTEXT\_LENGTH\_ERROR = Erreur de longueur sur une rubrique du buffer utilisateur du nœud référencé: {4} (value: {5})  
LOCAL\_REQUEST\_ALREADY\_EXIST = Requête externe déjà présente  
LOCAL\_REQUEST\_BAD\_USERBUFFER = Buffer utilisateur incorrect pour utiliser la requête externe ({6})  
LOCAL\_REQUEST\_NOT\_ACTIVE = Etat de la requête externe incorrect  
LOCAL\_UPDATE\_CURRENTLY\_POSTED = La modification de l'instance est déjà envoyée (data id: {3})  
LOCAL\_NO\_SERVER\_RESPONSE\_EXPECTED = Pas de réponse serveur attendue  
LOCAL\_LOCK\_SERVICE\_ALREADY\_REQUESTED = Service de Lock déjà présent dans la requête pour l'instance (data id: {3})  
LOCAL\_UNLOCK\_SERVICE\_ALREADY\_REQUESTED = Service de Unlock déjà présent dans la requête pour l'instance (data id: {3})  
LOCAL\_READ\_SERVICE\_ALREADY\_REQUESTED = Service de lecture identique déjà présent dans la requête  
LOCAL\_REQUEST\_BAD\_APPLICATION = Application eBusiness incorrecte utiliser la requête externe  
LOCAL\_REQUEST\_TOO\_LARGE = La requête a atteint son nombre maximum de services  
LOCAL\_CREATION\_CONTEXT\_INVALID = Impossible d'initialiser un contexte lorsqu'une requête externe existe

# 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 erroné (bib: {0}, ser: {1})  
SYSTEM\_MISPCV = Déphasage composants



# Index

## Actions COM

checkExistenceOfDependencies()	82
completeInstance	85
createInstance	50
createUserInstance()	56
deleteInstance	52
deleteUserInstance()	58
executeUserService()	83
getDetailFromData(LPDISPATCH d)	59
getDetailFromRowsIndex	60
getProxyClassName	91
getProxyContext	90
getReply(<nomDossier>ServerAction)	84
getUserDetailFromData({}LPDISPATCH d)	60
initFromProxyContext	91
is<CodeRubrique>Present	100
isReplyValid(<nomDossier>ServerAction)	85
lock	80
MainRequest	86, 87
MainRequest	45
modifyInstance	51
modifyUserInstance()	57
ping	86
ReadAllChildrenFromDetail()	75
readFirstChildrenFromDetail()	74
ReadInstance()	70, 71
readInstanceAndLock()	70
readNextPage()	77
readPreviousPage()	78
readWithAllChildren()	73
readWithAllChildrenAndLock()	73
readWithAllChildrenFrom({}Data d)	76
readWithFirstChildren()	72
readWithFirstChildrenAndLock()	72
readWithFirstChildrenFrom({}LPDISPATCH d)	75
resetAllRefreshOption()	63
resetCollection()	61
resetExtractMethodCodes()	62
resetSelectionCriteria()	64
resetSubSchema	67
resetUserRows()	63
resetUserServiceCodes()	62
selectInstances()	69
set<CodeRubrique>Present(BOOL Boolean Value)	100
setCheck(Long, Boolean)	100
transferReferenceFromSelectedRow({}LPDISPATCH d)	65
undoAllLocalFolderUpdates()	53
UndoAllLocalUpdate()	55
undoLocalUpdate	54
unlock()	81
updateFolder()	79

## Attributs COM

<CodeRubrique>	97
<CodeRubrique>ValidLabelsCount()	98
<CodeRubrique>ValidValuesCount()	97
accessInfoKey	26
accessInfoLabel	26
communicationResponseTime	42
detail	18
extractMethodCode	9
folderUserContext	28
GetErrorCount()	109
getGravity	110
getKey	109
getLabel	109
getType	109
globalSelection	14
host	33
lastReplyContext	40

localSort	16
location	32
locationsFile	34
LockTimestamp	25
manualCollectionReset	10
maximumReplyCount	41
maxNumberOfRequestedInstances	13
password	33
pendingReplyCount	41
port	34
referenceUserContext	28
refreshOption	12
serverAdapterName	35
serverCheckOption	11
serverResponseTime	42
setProperty	36
subSchema	44
userId	32
userServiceCode	24
VapError getElementAt(Int i)	108

## Classes

DataDescription	3
DataDescriptionUpdate	4
Schémas d'héritage de la classe ProxyLv	5
Schémas d'héritage des classes de données	3
SelectionCriteria	4
UserContext	4

## Événements COM

DEPENDENT_INSTANCES	96
FatalError	111
LocalError	110
LOCK_FAILED	95
LOCK_SUCCESSFUL	95
NO_DEPENDENT_INSTANCES	96
NO_PAGE_AFTER	93
NO_PAGE_BEFORE	93
NoError	110
NOT_FOUND	94
NOT_READ	94
PAGE_AFTER	93
PAGE_BEFORE	94
ServerError	110
SystemError	110

## Événements Java

dependentInstances	96
lockFailed	95
lockSuccessful	95
noDependentInstances	96
noPageAfter	93
noPageBefore	93
notFound	94
notRead	94
pageAfter	93
pageBefore	94

## Méthodes Java

<CodeRubrique>Error	98
<CodeRubrique>Index	100
belongsToSubschema	101
checkExistenceOfDependentInstances()	82
completeInstance	85
createInstance()	50
createUserInstance()	56
deleteInstance()	52
deleteUserInstance()	58
executeUserService()	83
getDetailFromDataDescription({}d)	59
getDetailFromRowsIndex	60
getFolderConstants()	66
getNodeConstants()	67

getProxyclassName.....	91	accessInfoKey .....	26
getProxyContext.....	90	accessInfoLabel .....	26
getReply(com.ibm.vap.generic.ServerActionContext s).....	84	asynchronous.....	40
getUserDetailFromDataDescription({} d) .....	60	communicationResponseTime.....	42
getXML .....	88	dataComparator .....	17
initFromProxyContext.....	91	detail .....	18
initializeInstance .....	64	extractMethodCode.....	9
ls< findDataFieldFormat .....	99	extractMethodCodes .....	8
ls< findDataFieldMaxLengtht .....	99	folderInstancesCount .....	29
ls< initializeFrom.....	99	folderUpdatedInstancesCount .....	29
is<CodeRubrique>Present .....	100	folderUserContext .....	28
isReplyValid(com.ibm.vap.generic.ServerActionContext aContext) .....	85	globalSelection .....	14
lock() .....	80	host .....	33
MainRequest .....	45, 86, 87	iRows.....	15
modifyInstance().....	51	iUpdatedFolders.....	19
modifyUserInstance() .....	57	iUpdatedInstances.....	20
ping .....	86	iUserInputRows .....	21
readAllChildren({}Data d).....	76	iUserOutputRows .....	22
readAllChildrenFromCurrentInstance().....	74	lastReplyContext.....	40
readFirstChildren({} d).....	75	localSort .....	16
readFirstChildrenFromCurrentInstance().....	74	location .....	32
readInstance() .....	70	locations .....	31
readInstanceAndLock() .....	70	lockTimestamp .....	25
readInstances) .....	71	manualCollectionReset.....	10
readInstanceWithAllChildren() .....	73	maximumNumberOfRequestedInstances .....	13
readInstanceWithAllChildrenAndLock() .....	73	maximumReplyCount .....	41
readInstanceWithFirstChildren().....	72	nodeUpdatedInstancesCount .....	30
readInstanceWithFirstChildrenAndLock().....	72	password .....	33
readNextPage().....	77	pendingReplyCount .....	41
readPreviousPage() .....	78	port .....	34
resetAllRefreshOption().....	63	property .....	36
resetCollection .....	61	referenceUserContext .....	28
resetExtractMethodCodes() .....	62	refreshOption .....	12
resetSelectionCriteria(s) .....	64	rows .....	15
resetSubSchema .....	67	selectionCriteria .....	7
resetUserRows() .....	63	serverAdapter .....	35
resetUserServiceCodes().....	62	serverAdapterName.....	35
restoreSelection ({}Data d) .....	61	serverCheckOption .....	11
selectInstances() .....	69	serverResponseTime .....	42
set<CodeRubrique>Present.....	100	subSchema .....	44
setCheck(int index, boolean a Boolean).....	100	subSchemaList.....	43
transferReferenceFromSelectedRow({} d) .....	65	tableModel.....	46
undoAllLocalFolderUpdates() .....	53	tableModel.....	47
undoAllLocalUpdate .....	55	updatedFolders.....	19
undoLocalFolderUpdates(ClientDataUpdate d) .....	52	updatedFoldersTableModel.....	46
undoLocalUpdate.....	54	updatedInstances.....	20
unlock().....	81	updatedInstancesTableModel .....	47
updateFolder().....	79	userDetail .....	23
updateFromXML.....	90	userId.....	32
<b>Propriétés Java</b>		userInputRows.....	21
<CodeRubrique> .....	97	userOutputRows.....	22
<CodeRubrique>Labels .....	98	userServiceCode .....	24
<CodeRubrique>Values .....	97	userServiceCodes .....	24