VisualAge Pacbase 2.5

Pacbench Client/Server

Graphic Clients - Reference Manual

Public Interface of Generated Components

DDOVI000255A

Note

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

According to your license agreement, you may consult or download the complete up-to-date collection of the VisualAge Pacbase documentation from the VisualAge Pacbase Support Center at:

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

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

5th Edition (November 1999)

This edition applies to the following licensed program:

- VisualAge Pacbase Version 2.5

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

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

NOTICES

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

Intellectual Property and Licensing
International Business Machines Corporation
North Castle Drive, Armonk, New-York 10504-1785
USA

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of information which has been exchanged, should contact:

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

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

IBM may change this publication, the product described herein, or both.

TRADEMARKS

IBM is a trademark of 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, and VisualAge are trademarks of International Business Machines Corporation, Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States and/or other countries.
Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.
UNIX is a registered trademark in the United States and/or other countries licensed exclusively through X/Open Company Limited.

All other company, product, and service names may be trademarks of their respective owners.

SUMMARY

A detailed Table of Contents follows this Summary.

Table of Contents

Foreword

CONTENTS OF THE MANUAL

This manual provides a comprehensive description of the Public Interface of components generated for the Graphic Clients of Pacbench C/S, according to three target environments: Java, Smalltalk and COM-compliant environments.

Each Proxy object's interface is generated using the characteristics –defined in VisualAge Pacbase– of a Logical View and its associated Business Component.

A Proxy object's public interface is made up of classes, characterized by a set of attributes or properties, actions or methods, and events. A GUI application handles these interface elements in order to manage each Logical View's processing according to its associated Business Component.

ORGANIZATION OF THE MANUAL

This manual is divided into chapters, followed by an *index*.

- Chapter **1**, *Classes*, on page 15, describes the classes of the public interface, including inheritance trees.
- Chapter **2**, *Attributes*, on page 19, provides the list of all attributes types existing for the various environments. For each attribute, the type (internal code), name (use name) and **`get`**/**`set`** are given.
- Chapter **3**, *Actions*, on page 58, describes the actions (called methods for the Java environment) –whether local or remote– with the declaration and use name.
- Chapter **4**, *Events*, on page 105, describes the Events and lists their codes.
- Chapter **5**, *Data Elements Handling Public Interface*, on page 113, documents the API for handling Data Elements.
- Chapter **6**, *Error Manager Public Interface [Smalltalk, COM]*, on page 116, describes the error management method for the Smalltalk and COM platforms.
- Chapter 7, *Management of Errors for Java Target*, on page 125, describes the error management for the Java platform only.

PREREQUISITES AND FURTHER READING

You should be familiar with the basic principles of Pacbench C/S. The explanations given in this manual assume you have such knowledge. For detailed information about these principles, see the *Pacbench C/S User's Guide, Vol. I: Concepts – Architectures – Environments.*

If you are new to this type of development, you may find it useful to read the *Pacbench C/S User's Guide, Vol. III: Graphic Clients*. This guide is designed to assist you in the development of Graphic Client components, through the presentation of various examples.

CONVENTIONS

The **`courier`** font signals any character string displayed or to be typed, as well as characters representing generated code.

The indication: “Internal Code” signals the code you will have to type in classic programming.

The indication: “Use Name” signals the corresponding label displayed in the Composition Editor, used in visual programming.

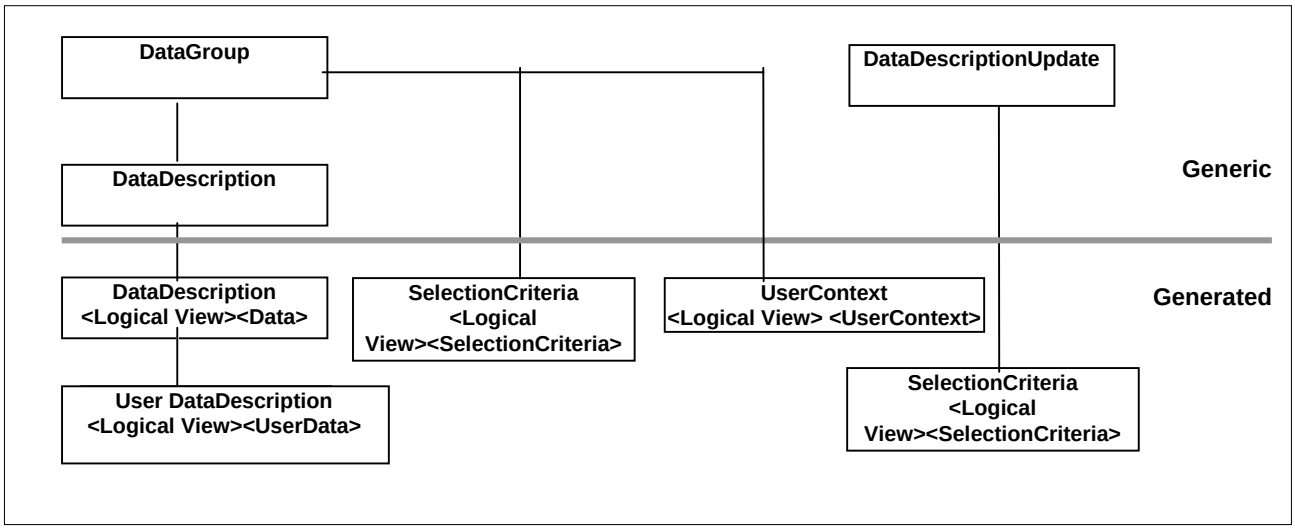
The indication **`[suffix]`** for Smalltalk signals the suffix selected for the **`ProxyLv`** class for each Business Component in VisualAge Pacbase (default suffix = **`ProxyLv`**).

1. Classes

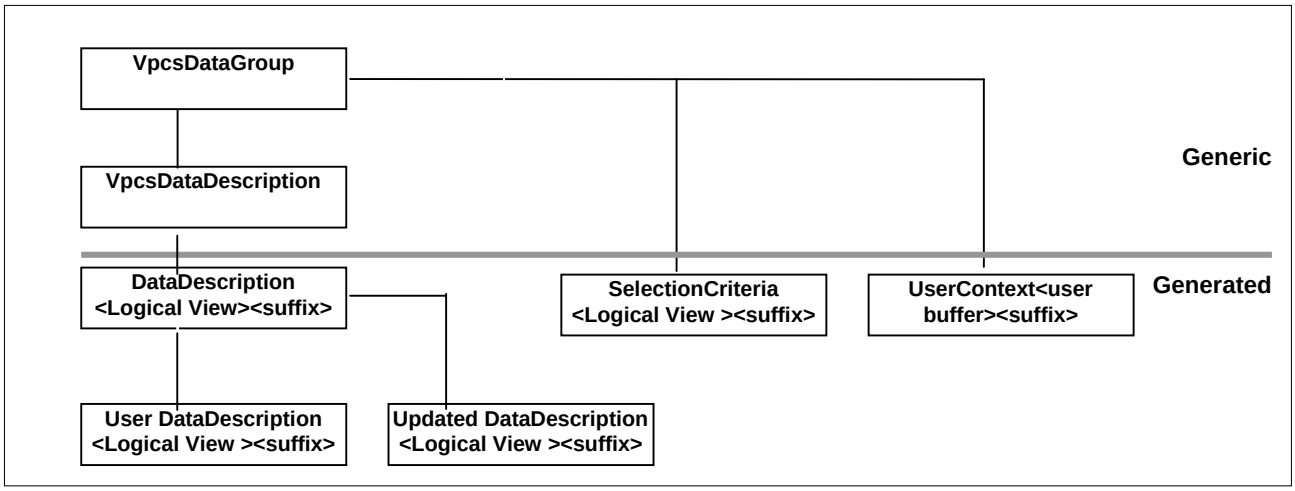
1.1. Data Classes

1.1.1. Inheritance Diagrams

1.1.1.1. Java and COM



1.1.1.2. Smalltalk



1.1.2. DataDescription Class

This class is generated for each Logical View Proxy or node of a Folder. It represents the structure of a Logical View, by defining one attribute for each identifier- or composition-type Data Element.

In the context of a Folder, the **`DataDescription`** class associated to a dependent node does not expose the identifier-type Data Elements of higher nodes.

An instance of this class corresponds to a Logical View instance handled by the application's graphic interface.

1.1.3. SelectionCriteria Class

This class is generated for each Logical View Proxy or node of a Folder. It represents the structure of the key and of the extraction parameters of a Logical View, by defining one attribute for each key- or extraction-parameter-type Data Element.

In the context of a Folder, the **`SelectionCriteria`** class associated to a dependent node does not expose the key-type Data Elements of nodes higher in the hierarchy.

There is only one instance of this class for each Logical View or node. This instance can be used to define the identifier –and possibly the extraction parameters– of the beginning of a collection (such as SelectInstance), or of a direct read (such as ReadInstance).

1.1.4. DataDescriptionUpdate Class

This class –also called **`UpdatedDataDescription`** in the Smalltalk environment– is generated for each Folder's root or dependent node. It represents the structure of a modified Logical View, and qualifies its modification type:

- **Created**: The Logical View associated to the node has been created locally.
- **Modified**: The Logical View associated to the node has been modified locally.
- **Deleted**: The Logical View associated to the node has been deleted locally.
- **Read**: At least one dependent instance in the Folder has been updated locally.

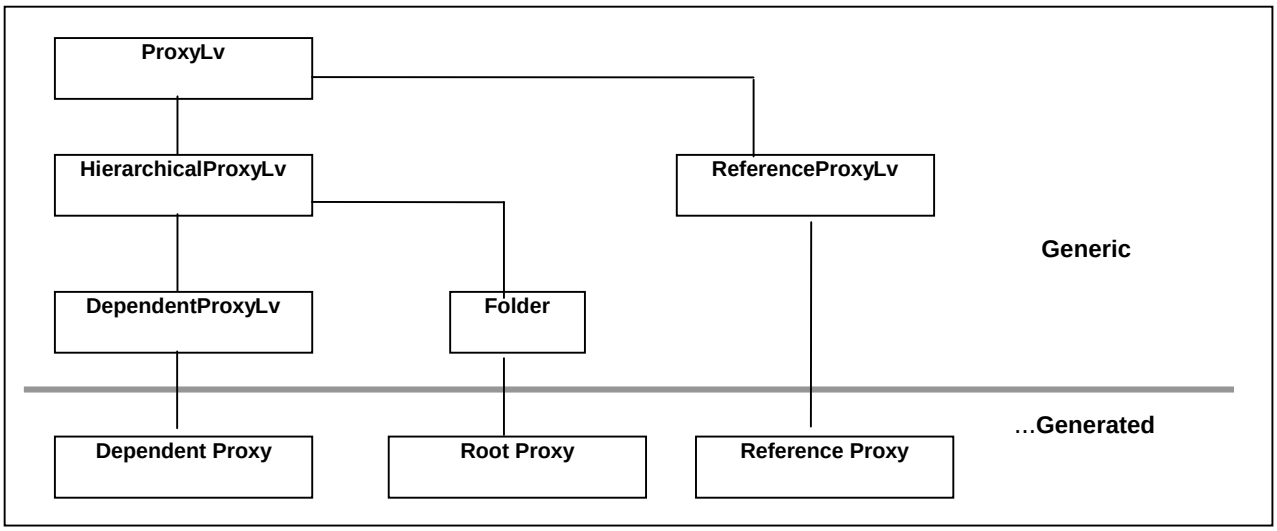
1.1.5. UserContext Class

This class is generated for each Logical or Folder View Proxy, or reference node of a Folder, when the servers used call a User Buffer. It represents the structure of the buffer, by defining an attribute for each Data Element.

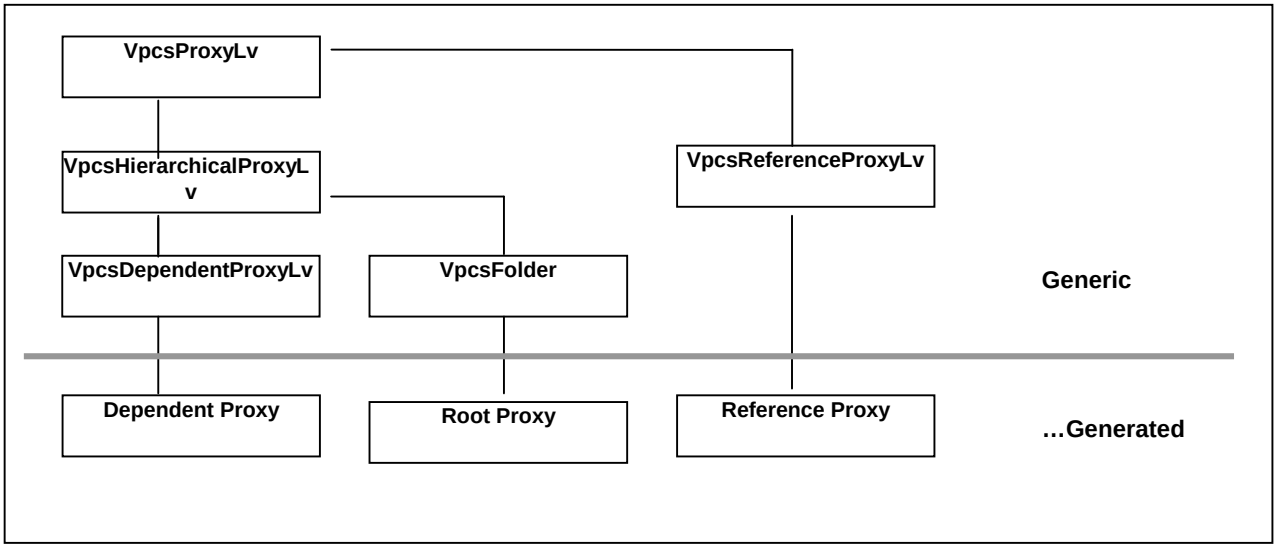
There is only one instance of this class. It is updated by the GUI or by replies from the remote server.

1.2. ProxyLv Class: Inheritance Diagrams

1.2.1. Java and COM



1.2.2. Smalltalk



2. Attributes

An attribute is linked to one of three types of elements making up the public interface of a class. For the public classes associated to a Proxy it may be a constant, a parameter or the result of an action. It is initialized by the application which uses the Proxy or by the Proxy itself, depending on the context.

2.1. Management of Selections

2.1.1. Selection Criteria

Description

This attribute defines all the Data Elements of the key- or extraction parameter-type, defined in the Logical View associated to a node.

For dependent nodes, key-type Data Elements already defined in the same attribute of the parent node are not exposed.

The description order is that defined in the Logical View.

Each Data Element exposed is set to an “empty” value, or to its default value if it is defined in the Repository.

This attribute is always available on root-, dependent-, and reference-type nodes.

It is available for read and write access.

Java

- Type `{Name of generated class}SelectionCriteria`
- Internal code `selectionCriteria`
- Use name `Selection Criteria`
- get/set `public <Type> selectionCriteria()` / set not available

Smalltalk

- Type `SelectionCriteria<LogicalViewCode>[suffix]`
- Internal code `SelectionCriteria<LogicalViewCode>[suffix]`
- Use name `selectionCriteria`
- get/set `selectionCriteria` / `selectionCriteria:`

COM

- Type `(Name of generated class) SelectionCriteria`
- Internal code `selectionCriteria`
- Use name `Selection Criteria`
- get/set C++ `public <Type> getSelectionCriteria()` / set not available

2.1.2. List of Available Extraction Methods

Description

This attribute exposes a list of extraction-method codes defined in the Business Component which manages the Logical View associated to the node.

It is available on root-, dependent-, and reference-type nodes, when at least one extraction method is defined in the Business Component managing the Logical View associated to the node.

It is available for read-only access.

Java

- Type `java.lang.String[]`
- Internal code `extractMethodCodes`
- Use name `extraction Method List`
- get/set `public String[] getExtractMethodCodes()` / set not available

Smalltalk

- Type `OrderedCollection` contenant `String`
- Internal code `extractMethodList`
- Use name `extractMethodList`
- get/set `extractMethodList` / set not available

COM

Available with a browsing API for collection-type attributes.

- Nb of elements `public Int getExtractMethodCodesCount()`
- Element `public String getExtractMethodCodesElementAt(Int i)`

2.1.3. Extraction Method to be Executed

Description

This attribute defines the extraction-method code to be implemented on a collection-selection action.

It may be set to an initialization value, if this is defined in the Proxy's Settings window.

It is available on root-, dependent-, and reference-type nodes, when at least one extraction action is defined in the Business Component managing the Logical View associated to the node.

It is available for read and write access.

Java

- Type `java.lang.String`
- Internal code `extractMethodCode`
- Use name `Extraction Method Code`
- get/set `public String getExtractMethodCode()` / `public void setExtractMethodCode(StringExtractMethodCode)`

Smalltalk

- Type `String`
- Internal code `extractMethodCode`
- Use name `extractMethodCode`
- get/set `extractMethodCode` / `extractMethodCode:`

COM

- Type `String`
- Internal code `extractMethodCode`
- Use name `extractMethodCode`
- get/set C++ `public String getExtractMethodCode()` / `public void setExtractMethodCode(String s)`

2.1.4. Management mode of the collection

Description

This attribute defines the collection management type, in return of a collection-selection action.

Two modes are available:

- Automatic management (default value)
- Manual management

The automatic management enables the replacement of the current collection with the selected instances, in return of a collection-selection action.

The manual management enables the completion of the current collection with the selected instances, in return of a collection-selection action. A selected instance, already present in the current collection, is refreshed if the instance of the current collection has not been locally modified.

The switch from one mode to another one does not lead to any change of the current collection.

As a default, the attribute is set to `false`.

In manual management, the pagination type for root- and reference-type nodes is always `extend`.

This attribute is systematically available on root-, dependent- or reference-type nodes.

It is available for read and write access.

Java

- Type `Boolean`
- Internal code `manualCollectionReset`
- Use name `manual Collection Reset`
- get/set
```
public Boolean isManualCollectionReset()
public void isManualCollectionReset(Boolean aBoolean)
```

Smalltalk

- Type `Boolean`
- Internal code `manualCollectionReset`
- Use name `manualCollectionReset`
- get/set `manualCollectionReset / manualCollectionReset:`

COM

- Type `Boolean`
- Internal code `manualCollectionReset`
- Use name `manualCollectionReset`
- get/set C++
```
public Boolean getManualCollectionReset()
public void setManualCollectionReset(Boolean b)
```

2.2. Management of Updates

2.2.1. Implementation of Server Data Checks

Description

This attribute triggers the data checks on the server when a server update method is being executed.

As a default, it is set to `false`. It may be set to another initialization value, if this is defined in the Proxy's Settings window.

It is available on root- and dependent-type nodes, when the associated Logical View is designed for update mode in its Business Component and the `CHECKSER` option of this Business Component is defined.

It is available for read and write access.

Java

- Type: `Boolean`
- Internal code: `serverCheckOption`
- Use name: `Server Check Option`
- get/set:
```
public Boolean getServerCheckOption()
public void setServerCheckOption(Boolean aBoolean)
```

Smalltalk

- Type: `Boolean`
- Internal code: `serverCheckOption`
- Use name: `serverCheckOption`
- get/set: `serverCheckOption` / `serverCheckOption:`

COM

- Type: `Boolean`
- Internal code: `serverCheckOption`
- Use name: `serverCheckOption`
- get/set C++:
```
public Boolean getServerCheckOption()
public void setServerCheckOption(Boolean b)
```

2.2.2. Server Data Refresh

Description

This attribute retrieves Logical View instances modified following an update performed by a Business Component. This function applies mainly to Logical Views with Data Elements calculated by the server.

As a default, this attribute is set to `false`. It may be set to another initialization value, if this is defined in the Proxy's Settings window.

It is available on root- and dependent-type nodes, when the associated Logical View is designed for update in its Business Component.

It is available for read and write access.

Java

- Type `Boolean`
- Internal code `refreshOption`
- Use name `Refresh Option`
- get/set
```
public Boolean getRefreshOption()
public void setRefreshOption(Boolean aBoolean)
```

Smalltalk

- Type `Boolean`
- Internal code `refreshOption`
- Use name `refreshOption`
- get/set `refreshOption` / `refreshOption:`

COM

- Type `Boolean`
- Internal code `refreshOption`
- Use name `refreshOption`
- get/set C++
```
public Boolean getRefreshOption()
public void setRefreshOption(Boolean b)
```

2.3. Exchange Flow Check

2.3.1. Limited Number of Exchanged Instances

Description

This attribute defines the maximum number of Logical View instances returned in one exchange by a Business Component upon a collection-retrieving action.

It is set to the Logical View's default iterative capacity. It may be set to another initialization value, if this is defined in the Proxy's Settings window.

It may be set to a value between 0 and n, where n may exceed the Logical View's iterative capacity. When this value is 0, simultaneous-selection actions on multiple nodes do not propagate reading requests to the current node.

This attribute is available on root- or reference-type nodes, and on dependent nodes with a maximum cardinal value of n.

It is available for read and write access.

Java

- Type: `Integer`
- Internal code: `maximumNumberOfRequestedInstances`
- Use name: `Maximum Number of Requested Instances`
- get/set:
```
public int getMaximumNumberOfRequestedInstances()
public void setMaximumNumberOfRequestedInstances(Int max)
```

Smalltalk

- Type: `Integer`
- Internal code: `maximumNumberOfRequestedInstances`
- Use name: `maximumNumberOfRequestedInstances`
- get/set:
```
maximumNumberOfRequestedInstances
maximumNumberOfRequestedInstances:
```

COM

- Type: `Integer`
- Internal code: `maxNumberOfRequestedInstances`
- Use name: `maxNumberOfRequestedInstances`
- get/set C++:
```
public int getMaxNumberOfRequestedInstances()
public void setMaxNumberOfRequestedInstances(Int i)
```

2.3.2. Unlimited Number of Exchanged Instances

Description

This attribute retrieves all the instances found in the database for the collection defined by the selection action. This function may generate a high number of exchanges between the client component and the server component.

As a default, it is set to `false`. It may be set to another initialization value, if this is defined in the Proxy's Settings window.

This attribute is available on root- and reference-type nodes, as well as dependent nodes with a maximum cardinal value of n.

It is available for read and write access.

Java

- Type `Boolean`
- Internal code `globalSelection`
- Use name `Global Selection`
- get/set
```
public Boolean getGlobalSelection()
public void setGlobalSelection(Boolean globalSelection)
```

Smalltalk

- Type `Boolean`
- Internal code `globalSelectionIndicator`
- Use name `globalSelectionIndicator`
- get/set `globalSelectionIndicator` / `globalSelectionIndicator:`

COM

- Type `Boolean`
- Internal code `globalSelection`
- Use name `globalSelection`
- get/set C++
```
public Boolean getGlobalSelection()
public void setGlobalSelection(Boolean b)
```

2.4. Logical View Instance Container

2.4.1. Instance List Presentation

Description

This attribute contains the current collection of the node to which it is associated. This collection is made of a set, or row, of Logical View instances. It results from the last reading action(s), and from local update actions performed on the node.

For a root node, the collection of instances exposed contains the Folders collection retrieved locally.

For a dependent node, the collection of instances exposed depends on the Logical View instance contained in the Instance-presentation attribute of its parent node. Other local instances which may have been retrieved locally are stored in the local cache and will be transferred according to the navigation operations performed in the Folder.

For a reference node the collection of instances exposed corresponds to the collection of Logical Views which can be referenced.

This attribute is available on root- or reference-type nodes, and on dependent nodes with a maximum cardinal value of n.

It is available for read-only access.

Java

- Type: `com.ibm.vap.generic.DataDescriptionVector` from generated `DataDescription`s
- Internal code: `rows`
- Use name: `Rows`
- get/set: `public DataDescriptionVector rows()`
  set not available

If the generation option "`Use IBM EAB classes`" is selected, another attribute is generated to simplify the use of IBM's EAB classes:

- Type: `COM.ibm.ivj.javabeans.IVector`
- Internal code: `iRows`
- Use name: `IRows`
- get/set: `public COM.ibm.ivj.javabeans.IVector iRows()`
  set not available

Smalltalk

- Type: `VpcsOrderedCollection` containing `DataDescription<LogicalViewCode>[suffix]`
- Internal code: `rows`
- Use name: `rows`
- get/set: `rows` / set not available

COM

Available with a browsing API for collection-type attributes.

- Nb of elements: `public Int getRowsCount()`
- Element: `public {Name of generated class}Data getRowsElementAt(Int i)`

2.4.2. Selection of Local or Server Criterion for Instance List Sort

Description

This attribute enables you to specify whether the collection contained in the instance list presentation attribute (page 26) is to be sorted according to the local sort criterion (`true`) or to the server sort criterion (`false`).

As a default, the instances stored in the Instance List Presentation attribute are sorted according to the local sort criterion.

This attribute is available on root-type or reference-type nodes, and on dependent nodes with a maximum cardinal value of n.

It is available for read and write access.

Java

- Type `Boolean`
- Internal code `localSort`
- Use name `localSort`
- get/set `public boolean getLocalSort()` / `public setLocalSort(boolean)`

Smalltalk

- Type `Boolean`
- Internal code `localSort`
- Use name `localSort`
- get/set `localSort` / `localSort:`

COM

Not available.

2.4.3. Local Criterion for Instance List Sort

Description

This attribute defines the local sort criterion applied to the collection stored in the Instance list presentation attribute (page 26).
As a default, the instances stored in the Instance List Presentation attribute are sorted in the order of the Logical View's key.
This attribute is available on root- or reference-type nodes, and on dependent nodes with a maximum cardinal value of n.
It is available for read and write access.

Java

In Java, the sort criterion is represented by the instance of a class implementing the generic interface `com.ibm.vap.generic.Comparator`. This interface is made of the following method declaration:

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

The implementation of this method must provide a sort criterion allowing the positioning of `a` before `b`, if `compare` returned a negative number, or `b` before `a` otherwise.

- Type `com.ibm.vap.generic.Comparator`
- Internal code `dataComparator`
- Use name -
- get/set
  ```
  public com.ibm.vap.generic.Comparator getDataComparator()
  public void setDataComparator(com.ibm.vap.generic.Comparator c)
  ```

Smalltalk

In Smalltalk, the following syntax is used to define the sort criterion:

`[ :a :b |` ***condition*** **`]`** where a and b represent two instances from the list of instances. The list of instances is thus sorted in order that all the couples of instances elaborated from the list match to the condition .

For instance, the default sort is realized by the following block: **`[ :a :b | a ident <= b ident ]`** where ident represents the Logical View identifier. The list is thus in the increasing order of the Logical View's identifiers.

- Type `BlockContextTemplate`
- Internal code `rowsSortBlock`
- Use name `rowsSortBlock`
- get/set `rowsSortBlock` / `rowsSortBlock:`

COM

Not available.

2.4.4. Instance Presentation

Description

This attribute is used to expose a particular Logical View instance (Detail). It defines all the Logical View's Data Elements which are not defined as extraction parameters.
For dependent nodes, Data Elements defining the key of parent node(s) are not exposed. Initialization of these Data Elements is automatically managed by the Folder View Proxy according to the current instances contained in the higher nodes.

When this attribute is empty, each Data Element exposed is set to an empty value, or to its default value if it is defined in the Repository.

After each direct reading or collection reading action returning only one instance, this attribute is set with the Logical View instance retrieved from the server.

This attribute is always available on root-, dependent-, and reference-type nodes.

It is available for read or write access.

Java

- Type `{Name of generated class}Data`
- Internal code `detail`
- Use name `Detail`
- get/set `public <Generated Type> detail()` / set not available

Smalltalk

- Type `DataDescription<LogicalViewCode>[suffix]`
- Internal code `detail`
- Use name `detail`
- get/set `detail` / `getDetailFromDataDescription: aDataDescription`

COM

- Type `{Name of generated class}Data`
- Internal code `detail`
- Use name `detail`
- get/set C++ `public <Type> getDetail()` / set not available

2.4.5. Presentation of Modified Folders

Description

This attribute exposes a list of locally modified Folders. It allows you, for example, to cancel local changes performed on a deleted Folder instance which does no longer appear in the Instance-list presentation attribute.

For each modified Folder, this attribute exposes:

- The Logical View instance of the Folder's root node.
- The number of modification services associated to the modified Folder instance.
- The modification status, which may be one of the following:

  #Created Locally created instance
  #Modified Locally modified instance
  #Deleted Locally deleted instance
  #Read Instance read, for which certain dependent instances have been updated locally.

This attribute is available on the root node of a Folder View Proxy when the options defined in the Business Components managing the Folder make the node modifiable.

It is available for read-only access.

Java

- Type `com.ibm.vap.generic.DataDescriptionUpdateVector` from `DataDescriptionUpdate`
- Internal code `updatedFolders`
- Use name `Updated Folders`
- get/set `public DataDescriptionUpdateVector updatedFolders()`
  set not available

If the generation option "`Use IBM EAB classes`" is selected, another attribute is generated to simplify the use of IBM's EAB classes:

- Type `COM.ibm.ivj.javabeans.IVector`
- Internal code `iUpdatedFolders`
- Use name `IUpdated Folders`
- get/set `public COM.ibm.ivj.javabeans.IVector iUpdatedFolders()`
  set not available

Smalltalk

- Type `VpcsOrderedCollection` containing `UpdatedDataDescription<LogicalViewCode>[suffix]`
- Internal code `updatedFolders`
- Use name `updatedFolders`
- get/set `updatedFolders` / set not available

COM

Available with a browsing API for collection-type attributes.

- Nb of elements `public Int getUpdatedFoldersCount()`
- Element `public {Name of generated class}DataUpdate getUpdatedFoldersElementAt(Int i)`

2.4.6. Presentation of Modified Instances

Description

This attribute exposes the list of the instances of the node which have been modified locally. It allows you, for example, to cancel local changes performed on a deleted instance which does no longer appear in the presentation attribute of a list of instances.

For each modified node, this attribute exposes:

- The Logical View instance of the node.
- The number of modification services associated with the modified instance.
- The modification status, which may be one of the following:

#Created	Locally created instance
#Modified	Locally modified instance
#Deleted	Locally deleted instance
#Read	Read instance, but some of its dependent instances have been updated locally.

This attribute is available on a root or dependent node of a Folder View Proxy if the Business Component associated with the node has an update service.

It is available for read-only access.

Java

- Type `com.ibm.vap.generic.DataDescriptionUpdateVector` de `DataDescriptionUpdate`
- Internal code `updatedInstances`
- Use name `Updated Instances`
- get/set `public DataDescriptionUpdateVector updatedInstances()`
  set not available

If the generation option "`Use IBM EAB classes`" is selected, another attribute is generated to simplify the use of IBM's EAB classes:

- Type `COM.ibm.ivj.javabeans.IVector`
- Internal code `iUpdatedInstances`
- Use name `IUpdated Instances`
- get/set `public COM.ibm.ivj.javabeans.IVector iUpdatedInstances()`
  set not available

Smalltalk

- Type `VpcsOrderedCollection` containing `UpdatedDataDescription<LogicalViewCode>[suffix]`
- Internal code `updatedInstances`
- Use name `updatedInstances`
- get/set `updatedInstances` / set not available

COM

Available with a browsing API for collection-type attributes

- Nb of elements `public Int getUpdatedInstancesCount()`
- Element `public {Name of generated class}DataUpdate getUpdatedInstancesElementAt(Int i)`

2.4.7. Presentation of Instances for a User Service

Description

This attribute contains a list of Logical View instances, created by local actions during the preparation of a User Service. These instances are independent from those contained in the Instance-list-presentation attribute (see page 26).

The rules for showing the instances according to the node-hierarchy do not apply to this attribute.

These instances will be sent to the server when executing the next server-type User Service submission.

This attribute is available on root- or dependent-type nodes, when at least one User Service is defined in the Business Component managing the Logical View, and the latter has an iterative capacity higher than 1.

This attribute is available for read-only access.

Java

- Type `com.ibm.vap.generic.DataDescriptionVector` from generated `UserDataDescription`
- Internal code `userInputRows`
- Use name `User Input Rows`
- get/set `public DataDescriptionVector userInputRows()`
  set not available

If the generation option "`Use IBM EAB classes`" is selected, another attribute is generated to simplify the use of IBM's EAB classes:

- Type `COM.ibm.ivj.javabeans.IVector`
- Internal code `iUserInputRows`
- Use name `IUser Input Rows`
- get/set `public COM.ibm.ivj.javabeans.IVector iUserInputRows()`
  set not available

Smalltalk

- Type `VpcsOrderedCollection` contenant `UserDataDescription<LogicalViewCode>[suffix]`
- Internal code `userServiceInputRows`
- Use name `userServiceInputRows`
- get/set `userServiceInputRows` / set not available

COM

Available with a browsing API for collection-type attributes.

- Nb of elements `public Int getUserInputRowsCount()`
- Element `public {Name of generated class}UserData getUserInputRowsElementAt(Int i)`

2.4.8. Presentation of Instances Returned by a User Service

Description

This attribute contains a list of Logical View instances, returned by the server-type User Service after its execution. These instances are independent from the instances contained in the Instance-list-presentation attribute (see page 26).

The rules for showing the instances according to the node-hierarchy do not apply to this attribute.

This attribute is available on root- or dependent-type nodes, when at least one User Service is defined in the Business Component managing the Logical View, and the latter has an iterative capacity higher than 1.

It is available for read and write access.

Java

- Type: `com.ibm.vap.generic.DataDescriptionVector` from generated `UserDataDescription`
- Internal code: `userOutputRows`
- Use name: `User Output Rows`
- get/set: `public DataDescriptionVector userOutputRows()`
  set not available

If the generation option "`Use IBM EAB classes`" is selected, another attribute is generated to simplify the use of IBM's EAB classes:

- Type: `COM.ibm.ivj.javabeans.IVector`
- Internal code: `iUserOutputRows`
- Use name: `IUser Output Rows`
- get/set: `public COM.ibm.ivj.javabeans.IVector iUserOutputRows()`
  set not available

Smalltalk

- Type: `VpcsOrderedCollection` containing `UserDataDescription<LogicalViewCode>[suffix]`
- Internal code: `userServiceOutputRows`
- Use name: `userServiceOutputRows`
- get/set: `userServiceOutputRows` / `userServiceOutputRows:`

COM

Available with a browsing API for collection-type attributes.

- Nb of elements: `public Int getUserOutputRowsCount()`
- Element: `public {Name of generated class}UserData getUserOutputRowsElementAt(Int i)`

2.4.9. Presentation of an Instance Linked to a User Service

Description

This attribute exposes a Logical View instance to be transmitted, or a Logical View instance returned by a server-type User Service. It defines all the Logical View's Data Elements which are not defined as extraction parameters.

The rules for showing the instances according to the node-hierarchy do not apply to this attribute. Consequently, in a dependent node, Data Elements defining the key of parent node(s) are exposed.

When this attribute is empty, each Data Element exposed is set to an empty value, or to its default value if it is defined in the Repository.

When a server-type User Service returns only one Logical View instance, it is exposed automatically by this attribute.

This attribute is available on root- or dependent-type nodes, when at least one User Service is defined in the Business Component managing the Logical View associated to the node.

It is available for read-only access.

Java

- Type `{Name of generated class}UserDataDescription`
- Internal code `userDetail`
- Use name `User Detail`
- get/set `public {}UserData userDetail()`
  set not available

Smalltalk

- Type `UserDataDescription<LogicalViewCode>[suffixe]`
- Internal code `userDetail`
- Use name `userDetail`
- get/set `userDetail` / set not available

COM

- Type `{Name of generated class}UserData`
- Internal code `userDetail`
- Use name `UserDetail`
- get/set C++ `public <Type> getUserDetail()` / set not available

2.5. Management of User Services

2.5.1. List of Available User Services

Description

This attribute exposes the list of User Services codes defined in the server managing the Logical View associated to the node.

This attribute is available on root- or dependent-type nodes, when at least one User Service is defined in the Business Component managing the Logical View associated to the node.

It is available for read-only access.

Java

- Type `java.lang.String[]`
- Internal code `userServiceCodes`
- Use name `User Service Codes`
- get/set `public String[] getUserServiceCodes()`
  set not available

Smalltalk

- Type `OrderedCollection` containing `String`
- Internal code `userServiceList`
- Use name `userServiceList`
- get/set `userServiceList` / not available

COM

Available with a browsing API for collection-type attributes.

- Nb of elements `public Int getUserServiceCodesCount()`
- Element `public String getUserServiceCodesElementAt(Int i)`

2.5.2. User Service to be Executed

Description

This attribute defines the code of the User Service which will be processed on the server managing the node when the User Services execution is triggered on a Folder's root node.

This attribute is available on root- or dependent-type nodes, when at least one User Service is defined in the Business Component managing the Logical View associated to the node.

It is available for read and write access.

Java

- Type `java.lang.String`
- Internal code `userServiceCode`
- Use name `User Service Code`
- get/set
```
public String getUserServiceCode()
public void setUserServiceCode(String code)
```

Smalltalk

- Type `String`
- Internal code `userServiceCode`
- Use name `userServiceCode`
- get/set `userServiceCode` / `userServiceCode:`

COM

- Type `String`
- Iinternal code `userServiceCode`
- Use name `userServiceCode`
- get/set C++
```
public String getUserServiceCode()
public void setUserServiceCode(String s)
```

2.6. Management of Logical Locks

2.6.1. Folder Lock Identifier

Description

This attribute exposes a character string calculated by the server and returned as a result of the last successful request of logical lock on a Folder instance. It is associated to the Folder instance currently contained in the root node.

This attribute is available on the root node of a Folder View Proxy when the 'logical locking' option of the Folder is set.

When the logical locking option of a Folder is set, and this attribute is empty, local updates on any of the Folder's nodes are denied.

This attribute is automatically set to an empty value for all Folder instances affected by a successful server update, or when an explicit logical unlocking action has been executed.

This attribute is available for read-only access.

No event is sent when the value of this attribute changes. Therefore, it is advised not to use attribute-to-attribute or event-to-method connections with this attribute.

Java

- Type `String`
- Internal code `lockTimestamp`
- Use name `Lock Timestamp`
- get/set `public String getLockTimestamp()`
  set not available

Smalltalk

- Type `String`
- Internal code `lockTimeStamp`
- Use name `lockTimeStamp`
- get/set `lockTimeStamp` / set not available

COM

- Type `String`
- Internal code `getLockTimestamp`
- Use name `lockTimestamp`
- get/set C++ `public String getLockTimestamp()` / set not available

2.7. Management of Selection-Return Messages

2.7.1. Message Label

Description

This attribute exposes the text of an information message returned by a server after execution of a selection action, when the end of the requested collection is reached, or when a requested instance cannot be found or is incomplete.

This attribute is always available on all types of nodes.

It is available for read-only access.

Java

- Type `java.lang.String`
- Internal code `accessInfoLabel`
- Use name `Access Info Label`
- get/set `public String getAccessInfoLabel()`
  set not available

Smalltalk

- Type `String`
- Internal code `accessInfoLabel`
- Use name `accessInfoLabel`
- get/set `accessInfoLabel` / set not available

COM

- Type `String`
- Internal code `accessInfoLabel`
- Use name `accessInfoLabel`
- get/set C++ `public String getAccessInfoLabel()` / set not available

2.7.2. Message Key

Description

This attribute exposes the key of an information message returned by a server after execution of a selection action, when the end of the requested collection is reached, or when a requested instance cannot be found or is incomplete. This attribute is always available on all types of nodes.

It is available for read-only access.

Java

- Type `java.lang.String`
- Internal code `accessInfoKey`
- Use name `Access Info Key`
- get/set `public String getAccessInfoKey()`
  set not available

Smalltalk

- Type `String`
- Internal code `accessInfoKey`
- Use name `accessInfoKey`
- get/set `accessInfoKey` / set not available

COM

- Type `String`
- Internal code `accesInfoKey`
- Use name `accessInfoKey`
- get/set C++ `public String get AccessInfoKey` / set not available

2.8. Management of Error Messages

2.8.1. Error Object

Description

This attribute contains an error instance returned as a result of an action applied to one or several of the Folder's nodes.

This attribute is always available on the root node.

It is available for read and write access.

Java

Not available.
Errors are generated as exceptions raised by Proxies actions.

Smalltalk

- Type `VpcsErrorManager`
- Internal code `errorManager`
- Use name `errorManager`
- get/set `errorManager` / `errorManager:`

COM

Not available.

2.9. Management of Events

2.9.1. List of Events from the Last Server Action

Description

This attribute contains a table of integral constants representing the various events returned by the last server action.

This attribute is always available on the root node.

It is available for read-only access.

Java

Not available

Smalltalk

Not available.

COM

Available with a management API for a stack. The retrieval method for an event retrieves the first event from the stack and cancels the event from the stack.

- Nb of elements `public Int getServerEventsCount()`
- Element `public String popServerEvent()`

2.10. Management of Contextual Information

2.10.1. Contextual Information

Description

This attribute contains the Data Elements of a contextual information structure sent and received each time a server action associated to a root- or dependent-type node is executed.
This attribute is available when a User Buffer has been defined at the Folder level. It is available for read-only access.

Java

- Type `{Name of generated class}Buffer`
- Internal code `folderUserContext`
- Use name `Folder User Context`
- get/set `public {}Buffer folderUserContext()`
  set not available

Smalltalk

- Type `UserContext<user buffer code>[suffix]`
- Internal code `folderUserContext`
- Use name `folderUserContext`
- get/set `folderUserContext` / set not available

COM

- Type `{Name of generated class}Buffer`
- Internal code `folderUserContext`
- Use name `folderUserContext`
- get/set C++ `public <Type> getFolderUserContext()` / set not available

2.10.2. Contextual Information Associated to Reference Nodes

Description

This attribute contains the Data Elements of a contextual information structure sent and received each time a server action associated to a reference node is executed.

It is available when a User Buffer has been defined at the level of the Business Component which manages the reference node.

It is available for read-only access.

Java

- Type `{Name of generated class}Buffer`
- Internal code `referenceUserContext`
- Use name `Reference User Context`
- get/set `public {}Buffer referenceUserContext()`/ set not available

Smalltalk

- Type `UserContext<user buffer code>[suffix]`
- Internal code `referenceUserContext`
- Use name `referenceUserContext`
- get/set `referenceUserContext` / set not available

COM

- Type `{Name of generated class}Buffer`
- Internal code `referenceUserContext`
- Use name `referenceUserContext`
- get/set C++ `public <Type> getReferenceUserContext()` / set not available

2.11. Available Counters

2.11.1. Total Number of Local Instances

Description

This attribute contains the number of local instances stored in the Folder View Proxy's cache, for all nodes.
This attribute is always available on the root node.
It is available for read-only access.

Java

- Type `int`
- Internal code `folderInstancesCount`
- Use name `Folder Instances Count`
- get/set `public int getFolderInstancesCount()` / set not available

Smalltalk

- Type `Integer`
- Internal code `folderInstancesCount`
- Use name `folderInstancesCount`
- get/set `folderUpdatedInstancesCount` / set not available

COM

This information is not referenced as an attribute in the Proxy's API. It is available through the following method:

- get/set C++ `public int getFolderInstancesCount()`
  set not available

2.11.2. Total Number of Update Services

Description

This attribute contains the number of updates that will be performed on the various Logical View servers when the server-update execution action is next sent. This number applies to all modified Folder instances, for all nodes.
This attribute is available on the root node when at least one Logical View associated to one of the Folder's nodes is designed for update in the Business Component managing the Logical View.

This attribute is available for read-only access.

Java

- Type `int`
- Internal code `folderUpdatedInstancesCount`
- Use name `Folder Updated Instances Count`
- get/set `public int getFolderUpdatedInstancesCount()`/ set not available

Smalltalk

- Type `Integer`
- Internal code `folderUpdatedInstancesCount`
- Use name `folderUpdatedInstancesCount`
- get/set `folderUpdatedInstancesCount` / set not available

COM

This information is not referenced as an attribute in the Proxy API. It is available through the following method:

- get/set C++ `public int getFolderUpdatedInstancesCount()`
  set not available

2.11.3. Number of Update Services Associated to a Node

Description

This attribute contains the number of updates which will be performed on the server associated to the node when the server-update execution action is next sent.

This attribute is available on each root- and dependent-type node whose associated Logical View is designed for update in the Business Component managing the View.

It is available for read-only access.

Java

- Type `int`
- Internal code `nodeUpdatedInstancesCount`
- Use name `Node Updated Instances Count`
- get/set `public int getNodeUpdatedInstancesCount()`
  set not available

Smalltalk

- Type `Integer`
- Internal code `nodeUpdatedInstancesCount`
- Use name `nodeUpdatedInstancesCount`
- get/set `nodeUpdatedInstancesCount` / set not available

COM

This information is not referenced as an attribute in the Proxy API. It is available through the following method:

- get/set C++ `public int getNodeUpdatedInstancesCount()`
  set not available

2.11.4. Number of Logical View Instances Reserved for a User Service

Description

This attribute contains the number of Logical View instances, for all the Folder's nodes, that will be sent to the server by a User Services execution action.

This attribute is available on a root node when at least one User Service is defined on the Business Component managing the Logical View of the root node or one of its dependent nodes.

It is available for read-only access.

Java

Not available.

Smalltalk

- Type `Integer`
- Internal code `folderInputUserServiceInstancesCount`
- Use name `folderInputUserServiceInstancesCount`
- get/set `folderInputUserServiceInstancesCount` / set not available

COM

Not available.

2.11.5. Number of Logical View Instances Processed by a User Service

Description

This attribute contains the number of Logical View instances –for all nodes– returned by the servers as a result of a User Services execution action.

This attribute is available on a root node when at least one User Service is defined on the Business Component managing the Logical View of the root node or one of its dependent nodes.

It is available for read-only access.

Java

Not available.

Smalltalk

- Type `Integer`
- Internal code `folderOutputUserServiceInstancesCount`
- Use name `folderOutputUserServiceInstancesCount`
- get/set `folderOutputUserServiceInstancesCount` / set not available

COM

Not available.

2.12. Management of Communications

2.12.1. List of Available Platforms

Description

This attribute contains the list of logical codes (locations) of all the execution platforms available for a Folder.

This attribute is always available on the root node.

It is available for read or write access.

Java

When using the Java or COM version with a gateway, the first execution of `getLocations` for Java and `getLocationsCount` for COM (either directly or when first accessing the server) triggers the call of a specific service of the gateway, whose role is to return the list of location logical names associated to the Folder.

- Type `java.lang.String[]`
- Internal code `locations`
- Use name `Locations`
- get/set `public String[] getLocations()`
  set not available

Smalltalk

- Type `VpcsOrderedCollection` containing `VpcsLocation`
- Internal code `locationList`
- Use name `locationList`
- get/set `locationList` / `locationList:`

COM

Available with a browsing API for collection-type attributes.

- Nb of elements `public Int getLocationsCount()`
- Element `public String getLocationsElementAt(Int i)`

2.12.2. Platform Selected for a Query Execution

Description

This attribute contains the logical code (location) of the next service to be executed on the server.

The population existing in the local cache is not reset at a location modification. However, it is possible to cancel from the local cache all the instances of the node and its dependents with the resetCollection action; see page 75.

This attribute is always available on the root node.

It is available for read or write access.

Java

- Type `String`
- Internal code `location`
- Use name `Location`
- get/set
```
public String getLocation()
public void setLocation(String newLoc)
```

Smalltalk

- Type `VpcsLocation`
- Internal code `location`
- Use name `location`
- get/set `location` / `location:`

COM

- Type `String`
- Internal code `location`
- Use name `location`
- get/set C++
```
public String getLocation()
public void setLocation(String l)
```

2.12.3. Log-in User Code

Description

This attribute contains the user code required for logging in to the selected execution platform.

This attribute is always available on the root node.

It is available for read / write access.

Java

- Type `String`
- Internal code `userId`
- Use name `User Id`
- get/set
get not available
```
public void setUserId(String uid)
```

Smalltalk

- Type `VpcsUserIdentification`
- Internal code `userId`
- Use name `serId`
- get/set `userId` / `userId:`

COM

- Type `String`
- Internal code `userId`
- Use name `userId`
- get/set C++ get not available / `public void setUserId(String u)`

2.12.4. Log-in Password

Description

This attribute contains the password associated to the user code required for logging in to the selected execution platform.

This attribute is always available on the root node.

It is available for read / write access.

Java

- Type `String`
- Internal code `password`
- Use name `Password`
- get/set get not available
  `public void setPassword(String pwd)`

Smalltalk

- Type `VpcsUserIdentification`
- Internal code `userPassword`
- Use name `userPassword`
- get/set `userPassword` / `userPassword:`

COM

- Type `String`
- Internal code `password`
- Use name `password`
- get/set C++ get not available / `public void setPassword(String p)`

2.12.5. Name of the Machine Hosting the Java/VisualAge Pacbase Gateway

Description

This attribute contains the TCP-IP address of the computer hosting the communications manager used for transmitting the messages to elementary Business Components.

This attribute is always available on the root node.

When the gateway is a Java/VisualAge Pacbase gateway, this attribute MUST be set.

It is available for read or write access.

Java

- Type `String`
- Internal code `host`
- Use name `Host`
- get/set
```
public String getHost()
public void setHost(String host)
```

Smalltalk

Not available.

COM

- Type `String`
- Internal code `host`
- Use name `host`
- get/set C++
```
public String getHost()
public void setHost(String h)
```

2.12.6. IP Port Associated to the Communications Manager

Description

This attribute contains the TCP-IP port associated to the communications manager used for transmitting the messages to elementary Business Components.

This attribute is always available on the root node.

The port must be the same as that used for the gateway. As a default it is set to 5647 on both sides.

It is available for read or write access.

Java

- Type `int`
- Internal code `port`
- Use name `Port`
- get/set
```
public int getPort()
public void setPort(int port)
```

Smalltalk

Not available.

COM

- Type `int`
- Internal code `port`
- Use name `port`
- get/set C++
```
public int getPort()
public void setPort(int p)
```

2.12.7. Setting of the Platforms File's Address

Description

This attribute contains the complete path of the platforms file used by the communications manager to determine the characteristics of the communication protocol providing access to an elementary Business Component.

This attribute is always available on the root node.

It should be used only when the Java application accesses the middleware locally, not via a gateway.

It is available for read or write access.

Java

This information is not referenced as an attribute in the Proxy's API. It is possible to modify it by executing the following method:

- get/set — No get
  `public void setLocationsFile(String f)`

Smalltalk

Not available.

COM

- Type — `String`
- Internal code — `locationsFile`
- Use name — `LocationsFile`
- get/set C++ — get not available

2.12.8. Selection of the Communication Adapter

Description

This attribute allows the definition of the communication mode used (`MiddlewareAdapter`, `GatewayAdapter`, etc..), indicating the name of the `ServerAdapter` class selected or, for Java target, passing an instance of the `ServerAdapter` selected.

This attribute is systematically available on the root node.

Java

- Type — `ServerAdapterName`
- Internal code — `serverAdapterName`
- Use name — `Server Adapter Name`
- get/set —
  ```
  public String getServerAdapterName ()
  public void setServerAdapterName(String className)
  ```
- Type — `ServerAdapter`
- Internal code — `serverAdapter`
- Use name — `Server Adapter`
- get/set —
  ```
  public ServerAdapter getServerAdapter ()
  public void setServerAdapter(ServerAdapter serverAdapter)
  ```

Smalltalk

Not available.

COM

- Type `String`
- Internal code `serverAdapterName`
- Use name `serverAdapterName`
- get/set
```
public String getServerAdapterName ()
public void setServerAdapterName(String className)
```

2.12.9. Management of Communication Parameters

Description

The « communication parameters » attributes are read and assigned using the `getProperty/setProperty` methods. These attributes define especially the required parameters to carry out a communication with the Server components according to the communication mode used (`MiddlewareAdapter`, `GatewayAdapter`, etc.).

The following list is an exhaustive list of the communication parameters:

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

This attribute is systematically available on the root node.

Java

- Type `Object`
- Internal code `property`
- Use name `Property`
- get/set
```
public Object getProperty (String attribut_name)/
public void setProperty(String attribut_name, Object
attribut_value)
```

Smalltalk

Not available.

COM

- Type `Int Double String`
- Internal code `setDoubleProperty setIntProperty setStringProperty`
- Use name `setDoubleProperty setIntProperty setStringProperty`
- get/set get not available /
```
public void setIntProperty(String attribut_name, Int
value)
public void setDoubleProperty(String attribut_name,
Double value)
public void setStringProperty(String attribut_name,
String value)
```

2.13. Management of Asynchronous Conversations

2.13.1. Determining the Type of Conversation

Description

This attribute is a Boolean value defining the current conversation type of the Folder. It must de set to `true` for recognition of an asynchronous-type conversation, to `false` for a synchronous-type conversation. As a default, it is set to `false`.
This attribute is always available on the root node.

It is available for read / write access.

Java

- Type `Boolean`
- Internal code `asynchronous`
- Use name `Asynchronous Mode`
- get/set
```
public Boolean isAsynchronous()
public void setAsynchronous(Boolean isAsynchronous)
```

Smalltalk

- Type `Boolean`
- Internal code `isAsynchronous:`
- Use name `isAsynchronous:`
- get/set `isAsynchronous` / `isAsynchronous:`

COM

- Type `Boolean`
- Internal code `asynchronous`
- Use name `asynchronous`
- get/set C++
```
public Boolean getAsynchronous()
public void setAsynchronous(Boolean a)
```

2.13.2. Last Identifier of an Asynchronous Conversation

Description

This attribute contains the identifier of the reply for last query performed with an asynchronous-type conversation on the current location.

This attribute is always available on the root node.

It is available for read access.

Java

- Type `com.ibm.vap.generic.ServerActionContext`
- Internal code `lastReplyContext`
- Use name `Last Reply Context`
- get/set `public ServerActionContext getLastReplyContext()`
  set not available

Smalltalk

- Type `VapServerActionContext`
- Internal code `lastReplyId`
- Use name `lastReplyId`
- get/set `lastReplyId` / set not available

COM

- Type `ServerActionContext`
- Internal code `lastReplyContext`
- Use name `lastReplyContext`
- get/set C++ `public ServerActionContext getLastReplyContext()`
  set not available

2.13.3. Maximum Number of Pending Replies

Description

This attribute contains the maximum number of reply-pending queries for the current location. This number is a specific parameter (`MWMAXREPLY`) of the asynchronous conversations, specified in the platforms file.

This attribute is always available on the root node.

It is available for read / write access.

Java

- Type `int`
- Internal code `maximumReplyCount`
- Use name `Maximum Reply Count`
- get/set
```
public int getMaximumReplyCount()
public void setMaximumReplyCount(int maximumReplyCount)
```

Smalltalk

- Type `Integer`
- Internal code `maximumReplyCount`
- Use name `maximumReplyCount`
- get/set `maximumReplyCount` / `maximumReplyCount:`

COM

- Type `int`
- Internal code `maximumReplyCount`
- Use name `maximumReplyCount`
- get/set C++ `public int getMaximumReplyCount()`
  set not available

2.13.4. Number of Pending Replies

Description

This attribute contains the number of asynchronous replies pending for a Folder. It is set to zero after each location change.

It is incremented when executing any query using an asynchronous-type conversation, except for update-type queries.

It is decremented after each return of a reply, or when pending queries are canceled.

This attribute is always available on the root node.

It is available for read / write access.

Java

- Type `int`
- Internal code `pendingReplyCount`
- Use name `Pending Reply Count`
- get/set `public int getPendingReplyCount()`
  set not available

Smalltalk

- Type `Integer`
- Internal code `pendingReplyCount`
- Use name `pendingReplyCount`
- get/set `pendingReplyCount` / set not available

COM

- Type `int`
- Internal code `pendingReplyCount`
- Use name `pendingReplyCount`
- get/set C++ `public int getPendingReplyCount()`
  set not available

2.14. Conversation Time

2.14.1. Communication Time

Description

This attribute contains the total communication time of the last conversation with the server.

It is set to an empty value.

This attribute is always available on the root node.

It is available for read access.

Java

Time, stated in milliseconds.

- Type `int`
- Internal code `communicationResponseTime`
- Use name `Communication Response Time`
- get/set `public int getCommunicationResponseTime()`
  set not available

Smalltalk

- Type `Time`
- Internal code `communicationResponseTime`
- Use name `communicationResponseTime`
- get/set `communicationResponseTime` / set not available

COM

- Type `int`
- Internal code `communicationResponseTime`
- Use name `communicationResponseTime`
- get/set C++ `public int getCommunicationResponseTime()`
  set not available

2.14.2. Execution Time of the Server Processing

Description

This attribute contains the total execution time of the server processing for the last conversation.

It is set to an empty value.

This attribute is always available on the root node.

It is available for read access.

Java

Time, stated in milliseconds.

- Type `int`
- Internal code `serverResponseTime`
- Use name `Server Response Time`
- get/set `public int getServerResponseTime()`
  set not available

Smalltalk

- Type `Time`
- Internal code `serverResponseTime`
- Use name `serverResponseTime`
- get/set `serverResponseTime` / set not available

COM

- Type `int`
- Internal code `serverResponseTime`
- Use name `serverResponseTime`
- get/set C++ `public int getServerResponseTime()`
  set not available

2.15. Sub-Schema Management

2.15.1. List of Available Sub-Schemas

Description

This attribute exposes the list of the sub-schemas available on the node. Sub-schemas are specified in the description of the Logical View associated with the node.

This attribute is available if the Business Components manage the presence of Data Elements (options **`VECTPRES=YES`** or **`CHECKSER=YES`**) and if the node includes at least one sub-schema.

It is available for read-only access.

Java

- Type **`subSchema[]`**
- Internal code **`subSchemaList`**
- Use name **`SubSchema List`**
- get/set **`public SubSchema[] getSubSchemaList()`**
  set not available

Smalltalk

- Type **`VpcsOrderedCollection`** containing **`VpcsSubSchema`**
- Internal code **`subSchemaList`**
- Use name **`subSchemaList`**
- get/set **`subSchemaList`** / set not available

COM

Available with a browsing API for collection-type attributes.

- Nb of elements **`public Int getSubSchemasCount()`**
- Element **`public String getSubSchemasElementAt(Int i)`**

2.15.2. Sub-schema to be Taken into Account

Description

This attribute contains the sub-schema to be taken into account when a selection, read or update action is performed.

Sub-schemas are specified in the description of the Logical View associated with the node.

This attribute is available if the Business Components manage the presence of Data Elements (options `VECTPRES=YES` or `CHECKSER=YES`) and if the node includes at least one sub-schema.

It is available for read and write access.

Java

- Type `SubSchema`
- Internal code `subSchema`
- Use name `Current SubSchema`
- get/set
```
public String getSubSchema()
public void setSubSchema(SubSchema SubSchema)
```

Smalltalk

- Type `VpcsSubSchema`
- Internal code `subSchema`
- Use name `subSchema`
- get/set `subSchema` / `subSchema:`

COM

- Type `String`
- Internal code `subSchema`
- Use name `subSchema`
- get/set C++ `String getSubSchema()` / `void setSubSchema(String s)`

2.16. Use of a JTable

2.16.1. Display of the Instances Collection in a JTable

Description

This attribute is available with Java only.

It enables you to insert a `JTable`, which is a swing component constitued of several rows and columns, and to display the collection of Logical View instances in this `JTable` via the `tableModel` attribute.

This attribute is available on all node types if you selected the generation option `Use Swing`.

It is available for read and write access.

Java

- Type `PacbaseTableModel`
- Internal code `tableModel`
- Use name `TableModel`

- get/set `public getTableModel`
  `public setTableModel(PacbaseTableModel)`

Smalltalk

Not available

COM

Not available

2.16.2. Display of the Updated Folders in a JTable

Description

This attribute is available with Java only.

It enables you to insert a `JTable`, which is a swing component constitued of several rows and columns, and to display the collection of updated folders in this `JTable` via the `updatedFoldersTableModel` attribute.

This attribute is available on all root-type nodes if you selected the generation option `Use Swing`.

It is available for read and write access.

Java

- Type `PacbaseUpdateTableModel`
- Internal code `updatedFoldersTableModel`
- Use name `updatedFoldersTableModel`
- get/set `public getUpdatedFoldersTableModel`
  `public`
  `setUpdatedFoldersTableModel(PacbaseUpdateTableModel)`

Smalltalk

Not available

COM

Not available

2.16.3. Display of the Updated Instances in a JTable

Description

This attribute is available with Java only.

It enables you to insert a `JTable`, which is a swing component constitued of several rows and columns, and to display the collection of updated instances in this `JTable` via the `updatedInstancesTableModel` attribute.

This attribute is available on all root-type nodes if you selected the generation option `Use Swing`.

It is available for read and write access.

Java

- Type `PacbaseUpdateTableModel`
- Internal code `updatedInstancesTableModel`
- Use name `updatedInstancesTableModel`
- get/set `public getUpdatedInstancesTableModel`
  `public setUpdatedInstancesTableModel(PacbaseUpdateTableModel)`

Smalltalk

Not available

COM

Not available

2.16.4. Display of the Instance Collection in input/output by a User Service in a JTable

Description

This attribute is available with Java only.

It enables you to insert a `JTable` in an application, which is a swing component constituted of several rows and columns and to display the collection of Logical View instances in input/output by a User Service in this `JTable` *via* the `tableModel` attribute.

This attribute is available on all node types if you have selected the generation option `Use Swing`.

It is available for read and write access.

Java

- Type `PacbaseTableModel`
- Internal code `tableModel`
- Use name `TableModel`
- get/set `public getUserInputTableModel`
  `public setUserInputTableModel(PacbaseTableModel)`
  `public getUserOutputTableModel`
  `public setUserOutputTableModel(PacbaseTableModel)`

Smalltalk

Not available

COM

Not available

3. Actions

An action is a piece of processing which can be executed by the Logical View Proxy. When an action requires some parameters for its execution, or when it returns results, those are passed through by the Logical View Proxy's attributes.

There are two types of actions for a Logical View Proxy:

- **Local actions,** which perform update operations on Logical View instances memorized by the Logical View Proxy.
- **Server actions,** which perform specific processing on the server. If the server uses a User Buffer, this type of action exchanges its contents every time it holds a conversation with the server.

Actions can therefore trigger either local processing internal, to the Logical View Proxy, or remote processing. These are standard selection and update processing actions, and user processing actions defined for the Business Component associated to the Logical View Proxy.

Note: The availability of these actions is indicated in the “Operation” paragraph for each action. In the case of a Java target, if an action is used although it is not available (wrong usage of the public method), a `java.lang.IllegalStateException` exception will be raised.

3.1. Actions Performed Locally

3.1.1. Updates

3.1.1.1. Creation of a Logical View Instance

Operation

This action creates a Logical View instance locally.

This action is valid if:

- The instance does not exist locally.
- Checks performed on all the instance's Data Elements did not return any errors.
- The parent instance of a dependent node is present locally.
- For a dependent node with a maximum cardinal value of 1, the created instance is the only one present locally for the parent instance (i.e., the parent instance has no dependent instance so far).
- The Folder has “modifiable” status.

If the action is valid:

- The “Total number of update services” counter is incremented by 1.
- The “Number of update services” counter associated to the node is incremented by 1.
- The “Total number of local instances” counter is incremented by 1.
- The new instance is included in the instance-list container associated to the node.
- The new modification is included in the modified Folders' presentation attributes.
- The no-error-detection event is sent.

If the action is not valid:

- The error is added to the Error Object.
- A local error event is sent.

This action is available if the Business Component allows for updates on the Logical View, and if all dependent nodes with a minimum cardinal value of 1 are present in the Folder View.

Java

- Declaration `public void createInstance() throws LocalException`
- Use name `Create Instance`

Smalltalk

- Declaration `createInstance`
- Use name `createInstance`

COM

- C++ declaration `public void createInstance()`
- Use name `createInstance`

3.1.1.2. Modification of a Logical View Instance

Operation

This action modifies a Logical View instance locally.

This action is valid if:

- The instance exists in the Instance-presentation attribute.
- The instance exists locally.
- Checks performed on all the instance's Data Elements did not return any errors.
- The Folder has “modifiable” status.

If the action is valid:

- The “Total number of update services” counter is incremented by 1 if no update transaction is currently associated to this instance.
- The “Number of update services” counter associated to the node is incremented by 1 if no update transaction is currently associated to this instance.
- The modification is included in the instance-list container associated to the node.
- The new modification is included in the modified Folders' presentation attributes.
- The no-error-detection event is sent.

If the action is not valid:

- The error is added to the Error Object.
- A local error event is sent.

This action is available if the Business Component allows for updates on the Logical View.

Java

- Declaration `public void modifyInstance() throws LocalException`
- Use name `Modify Instance`

Smalltalk

- Declaration `modifyInstance`
- Use name `modifyInstance`

COM

- C++ declaration `public void modifyInstance()`
- Use name `modifyInstance`

3.1.1.3. Deletion of a Logical View Instance

Operation

This action deletes a Logical View instance locally. It locally deletes all dependent nodes' instances one after the other.

This action is valid if:

- The instance exists locally.
- The instance exists in the Instance-presentation attribute.
- The parent instance of a dependent node is present locally.
- The Folder has “modifiable” status.

If the action is valid:

- The “Total number of update services” counter is incremented by 1 if no update transaction is currently associated to this instance.
- The “Number of update services” counter associated to the node is incremented by 1 if no update transaction is currently associated to this instance.
- The “Total number of local instances” counter is decremented by the number of dependent instances implicitly deleted, + 1.
- The instance is deleted from the instance-list container associated to the node.
- The new modification is included in the modified Folders' presentation attributes.
- All local instances which depend on the deleted instance are deleted.
- The no-error-detection event is sent.

If the action is not valid:

- The error is added to the Error Object.
- A local error event is sent.

This action is available if the Business Component allows for updates on the Logical View.

Java

- Declaration `public void deleteInstance() throws LocalException`
- Use name `Delete Instance`

Smalltalk

- Declaration `deleteInstance`
- Use name `deleteInstance`

COM

- C++ declaration `public void deleteInstance`
- Use name `deleteInstance`

3.1.2. Cancellation of Updates

1.1.1.1. Cancellation of a Folder's Updates

Operation

This action cancels all local updates performed on a Folder's instance, for all nodes, starting with the first local update.

This action is valid if:

- The instance exists in the root node's Instance-presentation attribute.

If the action is valid:

- The initial image of the instance and dependent instances is restored in the local cache, in the presentation attributes and in the instance list containers.
- The update-services total number counter is recalculated.
- The update-services number counter associated to the node is recalculated.
- The “Total number of local instances” counter is recalculated.
- The instance is deleted from the modified Folders' presentation attributes.
- The no-error-detection event is sent.

If the action is not valid:

- The error is added to the Error Object.
- A local error event is sent.

This action is available if at least one of the Business Components of the Folder allows for updates on a Logical View it manages.

Java

- Declaration `public void undoLocalFolderUpdates({Name of generated class} DataUpdate d)  throws LocalException`
- Use name `Undo Local Updates`

Smalltalk

- Declaration `undoLocalFolderUpdatesFor: UpdatedDataDescription{LogicalViewCode}{suffix}`
- Use name `undoLocalFolderUpdatesFor:`

COM

- Declaration C++ `public void undoLocalFolderUpdates({Name of generated class} DataUpdate d)`
- Use name `undoLocalFolderUpdates({Name of generated class} DataUpdate d)`

3.1.2.2. Cancellation of all Folders Updates

Operation

This action cancels all the local updates performed on all Folder instances, for all nodes.

Action impact:

- The initial images of all Folders instances and all their dependent instances are restored in the local cache, in the presentation attributes and in the instance-list containers.
- The “Total number of update services” counter is recalculated.
- The “Number of update services” counter associated to the node is recalculated.
- The “Total number of local instances” counter is recalculated.
- The modified Folders' presentation attributes are reset.
- The no-error-detection event is sent.

If the action is not valid:

- The error is added to the Error Object.
- A local error event is sent.

This action is available if at least one of the Business Components of the Folder allows for updates on a Logical View it manages.

Java

- Declaration `public void undoAllLocalFolderUpdates()`
- Use name `Undo all local folder updates`

Smalltalk

- Declaration `undoAllLocalFolderUpdates`
- Use name `undoAllLocalFolderUpdates`

COM

- C++ declaration `public void undoAllLocalFolderUpdates()`
- Use name `undoAllLocalFolderUpdates`

3.1.2.3. Cancellation of Updates on a Node Intance

Operation

This action cancels all local updates performed on an instance of the node, starting with the first local update. This action takes a node instance as a paramater.

This action is valid if:

- The instance passed as a parameter is a node instance which has been locally updated.

If the action is valid:

- The initial image of the instance and dependent instances (if the modification status of the node instance is not #Modified) is restored in the local cache, in the presentation attributes and in the instance-list containers.
- The counter of the total number of update services is recalculated.
- The counter of the update services number associated with the node is recalculated.
- The counter of the total number of local instances is recalculated.
- The instance and all its dependent instances are deleted from their respective presentation attributes of modified instances.
- The Folders' presentation attribute is updated.
- The no-error-detection event is sent.

If the action is not valid:

- The error is added to the Error Object.
- A local error event is sent.

This action is available on a root or dependent node of a Folder View Proxy if the Business Component associated with the node includes an update service.

Java

- Declaration
```
public void undoLocalUpdate({Name of generated class} DataUpdate d) throws LocalException
```
- Use name
```
Undo Local Update
```

Smalltalk

- Declaration
```
undoLocalUpdateFor: aDataDescription
```
- Use name
```
undoLocalUpdateFor:
```

COM

- C++ declaration
```
public void undoLocalUpdate({Name of generated class} DataUpdate d)
```
- Use name
```
undoLocalUpdate({Name of generated class} DataUpdate d)
```

3.1.2.4. Cancellation of Updates on all the Instances of a Node

Operation

This action cancels all the local updates performed on a node of all the instances of the current hierarchy, starting from the first local update.

Action impact:

- The initial images of the instances of the node for the current hierarchy and of all their dependent instances (if the modification status of a node instance is not #Modified) are restored in the local cache, in the presentation attributes and in the instance-list containers.
- The "total number of update services" counter is recalculated.
- The "number of update services" counter of the number of update services associated with the node is recalculated.
- The "total number of local instances" counter is recalculated.
- The instances and all their dependent instances are deleted from their respective presentation attribute of modified instances.
- The Folders' presentation attribute is updated.
- The no-error-detection event is sent.

This action is available on a root or dependent node of a Folder View Proxy if the Business Component associated with the node includes an update service.

Java

- Declaration `public void undoAllLocalUpdate()`
- Use name `Undo all local update`

Smalltalk

- Declaration `undoAllLocalUpdate`
- Use name `undoAllLocalUpdate`

COM

- C++ declaration `public void undoAllLocalUpdate()`
- Use name `undoAllLocalUpdate`

3.1.3. Management of User Services

3.1.3.1. Assignment of an Instance to a User Service

Operation

On a node, this action locally creates a new Logical View instance, reserved for the execution of the next User Service.

This action is valid if:

- The instance exists in the Instance-presentation attribute of an instance linked to a User Service.

If the action is valid:

- The counter of Logical View instances reserved for a User Service is incremented by 1.
- The instance is included in the presentation attributes of instances reserved for a User Service.
- The no-error-detection event is sent.

If the action is not valid:

- The error is added to the Error Object.
- A local error event is sent.

This action is available when the Business Component associated to the node manages at least one User Service.

Java

- Declaration `public void createUserInstance() throws LocalException`
- Use name `Create User Instance`

Smalltalk

- Declaration `createUserServiceInstance`
- Use name `createUserServiceInstance`

COM

- C++ declaration `public void createUserInstance()`
- Use name `createUserInstance`

3.1.3.2. Modification of an Assigned Instance

Operation

On a node, this action locally modifies a Logical View instance reserved for the execution of the next User Service.

This action is valid if:

- The instance exists in the presentation attribute of instances reserved for a User Service.
- The instance exists in the Instance-presentation attribute of an instance linked to a User Service.

If the action is valid:

- The modification is included in the presentation attribute of instances designed for a User Service.
- The no-error-detection event is sent.

If the action is not valid:

- The error is added to the Error Object.
- A local error event is sent.

This action is available when the Business Component associated to the node manages at least one User Service.

Java

- Declaration `public void modifyUserInstance() throws LocalException`
- Use name `Modify User Instance`

Smalltalk

- Declaration `modifyUserServiceInstance`
- Use name `modifyUserServiceInstance`

COM

- C++ declaration `public void modifyUserInstance()`
- Use name `modifyUserInstance`

3.1.3.3. Deletion of an Assigned Instance

Operation

On a node, this action locally deletes a Logical View instance reserved for the execution of the next User Service.

This action is valid if:

- The instance exists in the presentation attribute of instances reserved for a User Service.
- The instance exists in the presentation attribute of an instance linked to a User Service.

If the action is valid:

- The counter of Logical View instances reserved for a User Service is decremented by 1.
- The presentation attribute of instances designed for a User Service integrates the instance deletion.
- The no-error-detection event is sent.

If the action is not valid:

- The error is added to the Error Object.
- A local error event is sent.

This action is available when the Business Component associated to the node manages at least one User Service.

Java

- Declaration `public void deleteUserInstance() throws LocalException`
- Use name `Delete User Instance`

Smalltalk

- Declaration `deleteUserServiceInstance`
- Use name `deleteUserServiceInstance`

COM

- C++ declaration `public void deleteUserInstance()`
- Use name `deleteUserInstance`

3.1.4. Local Navigation in the Folders

Global Instance-presentation attributes setting rule:

When the instance-presentation attribute of a parent node contains a valid instance, the Instance- and Instance-list-presentation attributes of the dependent nodes are set according to the following rules:

- If the dependent node has a maximum cardinal values of n, it Instance-list-presentation attribute (Rows) is set with all the instances stored in the local cache and depending on the current instance of the parent node. If there is only one instance in the local cache, the Instance-presentation attribute (Detail) is also set with this instance.
- If the node has a maximum cardinal value of 1, its Instance-presentation attribute is set with the instance depending on the parent node's current instance, if it is found in the local cache.
- If the node does not meet any of the above rules, its Instance-presentation and Instance-list-presentation attributes are set to empty values.

3.1.4.1. Current Selection of an Instance in a Folder

Operation

This action assigns to the **Instance-presentation** attribute of a node an instance of the same type, such as, in particular, an instance from the **Instance-list-presentation** attribute.

This action is valid if:

- The input parameter for this action is an instance from `DataDescription`.

If the action is valid:

- The Instance-presentation attribute contains the instance to be assigned.
- The Instance- and Instance-list-presentation attributes of the dependent nodes are set according to the Global setting rule.
- The Folder's lock identifier is set if the instance to be assigned belongs to a root node which is currently locked.
- The no-error-detection event is sent.

If the action is not valid:

- The error is added to the Error Object.
- A local error event is sent.

This action is always available.

Java

- Declaration `public void getDetailFromDataDescription({}Data aData) throws LocalException`
- Use name `Get Detail From Data Description`

Smalltalk

- Declaration `getDetailFromDataDescription: aDataDescription`
- Use name `getDetailFromDataDescription:`

COM

- C++ declaration `public void getDetailFromData({Name of generated class}Data d)`
- Use name `getDetailFromData({Name of generated class}Data d)`

3.1.4.2. Selection of an Instance Associated to a User Service

Operation

This action assigns to the **Presentation of an instance for a User Service** attribute of a node an instance of the same type, such as, in particular, an instance from the **Presentation of an instance-list for a User Service** attribute.

Action impact:

- The presentation attribute of an instance reserved for a User Service contains the instance to be assigned.
- The no-error-detection event is sent.

If the action is not valid:

- The error is added to the Error Object.
- A local error event is sent.

This action is available when the Business Component associated to the node has at least one User Service.

Java

- Declaration `public void getUserDetailFromDataDescription({Name of generated class}Data d) throws LocalException`
- Use name `Get User Detail From Data Description`

Smalltalk

- Declaration `getUserDetailFromDataDescription:aDataDescription`
- Use name `getUserDetailFromDataDescription:`

COM

- C++ declaration `public void getUserDetailFromData({Name of generated class}Data d)`
- Use name `getUserDetailFromData({Name of generated class}Data d)`

3.1.5. Miscellaneous Initializations

3.1.5.1. Initialization of the collection

Operation

This action enables to discard the node instances and its dependent nodes from the local cache.

Action impact:

- the presentation attribute of a node instance is initialized to an empty value
- the list presentation attribute of a node instance is initialized to an empty value
- the presentation attribute of dependents nodes instance is initialized to an empty value
- the presentation attribute of dependents nodes instance lists is initialized to an empty value

The action is always available for all nodes.

Java

- Declaration `public void resetCollection ()`
- Nom d’utilisation `reset Collection`

Smalltalk

- Declaration `resetCollection`
- Nom d’utilisation `resetCollection`

COM

- C++ declaration `public void resetCollection()`
- Use name `resetCollection`

3.1.5.2. Initialization of Extraction Methods

Operation

This action sets the **Extraction method to be executed** attributes of the node and all its dependent nodes to empty values.

The “Extraction method to be executed” attribute of each affected node contains an empty value.

This action is always available for all nodes.

Java

- Declaration `public void resetExtractMethodCodes()`
- Use name `Reset Extract Method Codes`

Smalltalk

- Declaration `resetExtractMethodCodes`
- Use name `resetExtractMethodCodes`

COM

- C++ declaration `public void resetExtractMethodCodes()`
- Use name `resetExtractMethodCodes`

3.1.5.3. Initialization of the User Services

Operation

This action sets the **User Service to be executed** attributes of the node and all its dependent nodes to empty values.

The “User Service to be executed” attribute of each affected node contains an empty value.

This action is available when the Business Component associated to the node has at least one User Service.

Java

- Declaration `public void resetUserServiceCodes()`
- Use name `Reset User Service Codes`

Smalltalk

- Declaration `resetUserServiceCodes`
- Use name `resetUserServiceCodes`

COM

- C++ declaration `public void resetUserServiceCodes()`
- Use name `resetUserServiceCodes`

3.1.5.4. Initialization of the “Presentation of Instances for a User Service” Container

Operation

This action sets the **Presentation of instances for a User Service** attributes of the node and all its dependent nodes to empty values.

Action impact:

- For each affected node, the “Presentation of instances designed for a User Service to be executed” attribute contains an empty value.

This action is available when the Business Component associated to the node has at least one User Service.

Java

- Declaration `public void resetUserRows()`
- Use name `Reset User Rows`

Smalltalk

- Declaration `resetUserServiceInputInstances`
- Use name `resetUserServiceInputInstances`

COM

- C++ declaration `public void resetUserRows()`
- Use name `resetUserRows`

3.1.5.5. Initialization of the Update-Refresh Option

Operation

This action inhibits the update-refresh option on the node and on its dependent nodes, by setting the corresponding Boolean value to “false”.

This action is available when the Business Component associated to the node allows for updates on the Logical View it manages.

Java

- Declaration `public void resetAllRefreshOption()`
- Use name `Reset All Refresh Option`

Smalltalk

- Declaration `resetAllRefreshOption`
- Use name `resetAllRefreshOption`

COM

- C++ declaration `public void resetAllRefreshOption()`
- Use name `resetAllRefreshOption`

3.1.5.6. Initialization of Selection Criteria

Operation

This action sets the Selection Criteria attributes of the node and all its dependent nodes to empty values.

As a result of this action, the Selection Criteria attribute of each affected node contains an empty value.

This action is always available for all root- and dependent-type nodes.

Java

- Declaration `public void resetSelectionCriterias()`
- Use name `Reset Selection Criterias`

Smalltalk

- Declaration `resetSelectionCriterias`
- Use name `resetSelectionCriterias`

COM

- C++ declaration `public void resetSelectionCriterias()`
- Use name `resetSelectionCriterias`

3.1.5.7. Addition of instances in the local cache without server access

Operation

This action allows the addition, in the local cache, of non-read instances from the server. These instances have not the locally-created status.

This action is valid if the instance does not exist in local mode, whatever its status is.

If the action is valid:

- The “Total number of local instances” counter is incremented by 1.
- The instance list container associated to the node embeds the new instance
- The event ‘no error detected’ is sent.

If the action is not valid:

- The error is added to the Error Object.
- A local error event is sent.

This action is always available on all nodes.

Java

- Declaration `public void initializeInstance() throws LocalException`
- Use name `initializeInstance`

Smalltalk

Not available.

COM

Not available.

3.1.6. Management of Referenced Instances

3.1.6.1. Assignment of a Referenced Instance

Operation

This action maps the identifier-type Data Elements in a reference-node instance used as a parameter of the action to the ‘foreign key’-type Data Elements of the referencing-node instance.

This action is valid if:

- An instance exists in the Instance-presentation attribute of the referencing node.
- The reference-node's instance does not contain an empty value.

If the action is valid:

- The ‘foreign key’-type Data Elements in the Instance-presentation attribute of the referencing node are initialized with the identifier-type Data Elements of the reference node.

If the action is not valid:

- The error is added to the Error Object.
- A local error event is sent.

This action is always available on reference nodes.

Java

- Declaration `public void transferReferenceFromSelectedRow({Name of generated class}Data data) throws LocalException`
- Use name `Transfer Reference From Selected Row`

Smalltalk

- Declaration `transferReferenceFromSelectedRow: aRow`
- Use name `transferReferenceFromSelectedRow:`

COM

- C++ declaration `public void transferReferenceFromSelectedRow({Name of generated class}Data d)`
- Use name `transferReferenceFromSelectedRow({Name of generated class}Data d)`

3.1.7. Management of Collections Acquisition

3.1.7.1. Retrieval of the Event Associated to the Last Server Selection

Operation

This action retrieves the last event sent for a node during the last selection action.

As a result of this action, the event is sent (Presence or Absence of a preceding page, Presence or Absence of a following page, Reading on a record not found, No reading performed).

This action is always available for all root- and dependent-type nodes.

Java

Not useful in the context of ServerEventStack..

Smalltalk

- Declaration `getLastSelectResponseStatus`
- Use name `getLastSelectResponseStatus`

COM

Not useful in context of the `popServerEvent` method.

3.1.8. Retrieval of Proxies' Generation Contexts

3.1.8.1. Generation Context of a Folder

Operation

This action retrieves the VisualAge Pacbase constants from the Services Manager associated to the root node, in the form of a collection of character strings containing the following information:

- Services Manager external name
- VisualAge Pacbase code of the Folder (or Business Component)
- Database code of the VisualAge Pacbase Repository
- Library code
- Generation-session number
- User code
- Generation date
- Generation time
- Folder View code

This action is always available for a root node.

Java

- Declaration `public String[] getFolderConstants()`
- Use name `Get Folder Constants`

Smalltalk

- Declaration `getFolderConstants`
- Use name `getFolderConstants`

COM

Available with a browsing API for collection-type attributes.

- Nb of elements `public Int getFolderConstantsCount()`
- Element `public String getFolderConstantsElementAt(Int i)`

3.1.8.2. Generation Context of a Node

Operation

This action retrieves the VisualAge Pacbase constants from the Business Component associated to the node.

Action impact:

- The action returns a collection of character strings containing the following information:
  - External name of the Business Component
  - VisualAge Pacbase code of the Business Component
  - Database code of the VisualAge Pacbase Repository
  - Library code
  - Generation-session code
  - User code
  - Generation date
  - Generation time
  - Operations version of the Business Component

This action is always available for all types of nodes (root, dependent, or reference).

Java

- Declaration `public String[] getNodeConstants()`
- Use name `Get Node Constants`

Smalltalk

- Declaration `getNodeConstants`
- Use name `getNodeConstants`

COM

Available with a browsing API for collection-type attributes.

- Nb of elements `public Int getNodeConstantsCount()`
- Element `public String getNodeConstantsElementAt(Int i)`

3.1.9. Sub-Schema Management

3.1.9.1. No Selection of Sub-Schema

Operation

This action resets the **subSchema** attribute, that is no sub-schema is selected.

This action is available if the Business Components manage the presence of Data Elements (**VECTPRES=YES** or **CHECKSER=YES**) and if the node includes at least one sub-schema.

Java

- Declaration `public void resetSubSchema()`
- Use name `Reset SubSchema`

Smalltalk

- Declaration `resetSubSchema`
- Use name `resetSubSchema`

COM

- C++ declaration `public resetSubSchema()`
- Use name `ResetSubSchema`

3.1.9.2. Data Element Belonging to the Sub-Schema

Operation

This action enables you to know whether the Data Element passed as a parameter belongs to the sub-schema associated with the instance contained in the **detail** attribute.

This action is available if the Business Components manage the presence of Data Elements (**VECTPRES=YES** or **CHECKSER=YES**) and if the node includes at least one sub-schema.

Java

- Declaration `public boolean belongsToSubSchema(int DataElementindex)`
- Use name `Belongs to SubSchema`

Smalltalk

- Declaration `belongsToSubschema: <#codeRubrique>`
- Use name `belongsToSubschema:`

COM

- Declaration `belongsToSubSchema(int DataElementindex)`
- Use name `belongsToSubSchema`

3.2. Actions Performed on a Remote Server

Reminder: **Global Instance-presentation attributes setting rule**:

When the Instance-presentation attribute of a parent node contains a valid instance, the Instance- and Instance-list-presentation attributes of the dependent nodes are set according to the following rules:

- If the dependent node has a maximum cardinal value of n, its Instance-list-presentation attribute is set with all the instances stored in the local cache and dependent on the current instance of the parent node. If there is only one instance in the local cache, the Instance-presentation attribute is also set with this instance.
- If the node has a maximum cardinal value of 1, its Instance-presentation attribute is set with the instance dependent on the parent node's current instance, if it is found in the local cache.
- If the node does not meet the above-stated rules, its Instance-presentation and Instance-list-presentation attributes are set to empty values.

3.2.1. Selection on a Node

3.2.1.1. Selection of a Set of Instances

Operation

This action defines a Logical View instance collection associated to the node, and retrieves all of this collection's instances, or the first page.

If the action is valid:

- The Instance-list-presentation attribute is modified according to the value of the management mode attribute of the collection.
- The “Total number of local instances” counter is initialized.
- The selection-return message's label and key are initialized if the last instance of the collection has been retrieved.
- The Instance- and Instance-list presentation attributes of dependent-type nodes are set to empty values.
- A no-error-detection event is sent.
- The event “Presence of updatable local instances” is sent, with a collection management in automatic mode, if updatable instances are still present in the local cache.
- The event “Retrieval of a collection's first page” is sent if the Folder operates in `non extend` mode and with the collection management in automatic mode.
- The event “Presence of at least one following page” is sent if the collection's last instance has not been retrieved.
- The event “Retrieval of the last page” is sent if the last collection's instance has been retrieved.

If the action is not valid:

- The error is added to the Error Object.
- An error event is sent according to the type of error.

In all circumstances:

- Conversation time counters are set.
- Contextual information attributes –if they are present– are set.

This action is always available on root- and reference-type nodes.

Java

- Declaration `public void selectInstances() throws ServerException, CommunicationError, SystemError, LocalException`
- Use name `Select Instances`

Smalltalk

- Declaration `selectInstances`
- Use name `selectInstances`

COM

- C++ declaration `public void selectInstances()`
- Use name `selectInstances`

3.2.1.2. Reading of an Instance with or without Logical Locking

Operation

This action retrieves a Logical View instance associated to the node, and appropriates it for exclusive update, if relevant.

This action is valid if:

- The instance is not already locked, in the case of an action with locking.

If the action is valid:

- The Instance-presentation attribute is set.
- The Instance-list-presentation attribute is modified according to the value of the management mode attribute of the collection.
- The “Total number of local instances” counter is initialized.
- The selection-return message's label and key are initialized if the last instance of the collection has not been retrieved.
- The Instance- and Instance-list- presentation attributes of dependent nodes are set to empty values.
- A no-error-detection event is sent.
- The event “Presence of updatable local instances” is sent, with the collection management in automatic mode, if updatable instances are still present in the local cache.
- The Folder-lock identifier is initialized in the case of a locking request.

If the action is not valid:

- The error is added to the Error Object.
- An error event is sent according to the type of error.
- The Folder-lock identifier is set to an empty value in the case of a locking request, and the Folder changes to the ‘not-modifiable’ status.

In all circumstances:

- Conversation time counters are set.
- Contextual information attributes –if they are present– are set.

This action is always available on all nodes.

Java

- Declaration

```
public void readInstance() throws LocalException,
ServerException, CommunicationError, SystemError
public void readInstanceAndLock() throws LocalException,
ServerException, CommunicationError, SystemError
```

- Use name

```
Read Instance
Read Instance And Lock
```

Smalltalk

- Declaration

```
readInstance[AndLock]
```

- Use name

```
readInstance[AndLock]
```

COM

- C++declaration

```
public void readInstance()
public void readInstanceAndLock()
```

- Use name

```
readInstance
readInstanceAndLock
```

3.2.2. Concurrent Selection on Multiple Nodes with or without Locking

3.2.2.1. Reading of an Instance and its Immediate Hierarchy

Operation

This action retrieves a Logical View instance associated to the node, and appropriates it for exclusive update –if relevant, then retrieves all or part of the instances of first-level dependent nodes.

This action is valid if:

- The instance is not locked, in the case of a action with locking.

If the action is valid:

- The Instance-presentation attribute is initialized with the result of the selection.
- The Instance-list-presentation attribute is modified according to the value of the management mode attribute of the collection.
- The “Total number of local instances” counter is initialized.
- The selection-return message's label and key are initialized for each first-level dependent node if the collection's last instance has been retrieved.
- The Instance- and Instance-list-presentation attributes of first-level dependent nodes in the hierarchy are initialized with the result of the selection, except for those for which the number of exchanged instances was set to zero (they are initialized to empty values). For the Instance-list attribute, the modification is made according to the value of the collection management mode attribute, the Instance-list attribute being associated to each node.
- The Instance- and Instance-list-presentation attributes of dependent nodes of a hierarchical level higher than one are set to empty values.

- A no-error-detection event is sent.
- The event “Presence of updatable local instances” is sent, with the collection management in automatic mode, if updatable instances are still present in the local cache.
- A “Record not found” event is sent on every node participating in the selection and having a maximum cardinal value of 1, and whose instance has not been retrieved.
- The Folder-lock identifier is set to an empty value in the case of a locking request.

If the action is not valid:

- The error is added to the Error Object.
- An error event is sent according to the type of error.
- The Folder-lock identifier is set to an empty value in the case of a locking request, and the Folder changes to the ‘not-modifiable’ status.

In all circumstances:

- Conversation time counters are set.
- Contextual information attributes –if they are present– are set.

This action is always available on all root- and dependent-type nodes.

Java

- Declaration

```
public void readInstanceWithFirstChildren() throws
LocalException, ServerException, CommunicationError,
SystemError
public void readInstanceWithFirstChildrenAndLock() throws
LocalException, ServerException, CommunicationError,
SystemError
```

- Use name

```
Read Instance With First Children
Read Instance With First Children And Lock
```

Smalltalk

- Declaration

```
readInstanceWithFirstChildren[AndLock]
```

- Use name

```
readInstanceWithFirstChildren[AndLock]
```

COM

- C++ declaration

```
public void readWithFirstChildren()
public void readWithFirstChildrenAndLock()
```

- Use name

```
readWithFirstChildren
readWithFirstChildrenAndLock
```

3.2.2.2. Reading of an Instance and its Complete Hierarchy

Operation

This action retrieves a Logical View instance associated to the root node, appropriates it for exclusive update if relevant, and retrieves all the instances of each dependent node, whatever its hierarchical level.

This action is valid if:

- The instance is not already locked, in the case of a action with locking.

If the action is valid:

- The Instance-presentation attribute of the root node is initialized with the selection result.
- The Instance-list-presentation attribute of the root node is modified according to the value of the collection management mode attribute.
- The “Total number of local instances” counter is initialized.
- The selection-return message's label and key are initialized for each dependent node if the collection's last instance has been retrieved.
- The Instance- and Instance-list-presentation attributes of a dependent node are set according to the Global setting rule. For the Instance-list attribute, the modification is made according to the value of the collection management mode attribute, the Instance-list attribute being associated to each node.
- A no-error-detection event is sent.
- The event “Presence of updatable local instances” is sent, with the collection management in automatic mode, if updatable instances are still present in the local cache.
- A “Record not found” event is sent on every node participating in the selection and having a maximum cardinal value of 1, and whose instance has not been retrieved.
- The Folder-lock identifier is set to an empty value in the case of a locking request.

If the action is not valid:

- The error is added to the Error Object.
- An error event is sent according to the type of error.
- The Folder-lock identifier is set to an empty value in the case of a locking request, and the Folder changes to the ‘not-modifiable’ status.

In all circumstances:

- Conversation time counters are set.
- Contextual information attributes –if they are present– are set.

This action is always available on root nodes.

Java

- Declaration
```
public void readInstanceWithAllChildren() throws
LocalException, ServerException, CommunicationError,
SystemError
public void readInstanceWithAllChildrenAndLock() throws
LocalException, ServerException, CommunicationError,
SystemError
```
- Use name
```
Read Instance With All Children
Read Instance With All Children And Lock
```

Smalltalk

- Declaration
```
readInstanceWithAllChildren[AndLock]
```
- Use name
```
readInstanceWithAllChildren[AndLock]
```

COM

- C++ declaration
```
public void readWithAllChildren()
public void readWithAllChildrenAndLock()
```
- Use name
```
readWithAllChildren
readWithAllChildrenAndLock
```

3.2.2.3. Reading of the Immediate Hierarchy of a Current Instance

Operation

For a node, this action retrieves all or part of the instances of first-level dependent nodes, depending on the instance found in the node's **Instance-presentation** attribute.

This action is valid if:

- The node's Instance-presentation attribute contains an instance.

If this action is valid, its result is the same as that of a read action on an instance and its immediate hierarchy.

This action is always available on all root- and dependent-type nodes.

Java

- Declaration
```
public void readFirstChildrenFromCurrentInstance() throws
LocalException, ServerException, CommunicationError,
SystemError
```
- Use name
```
Read First Children From Detail
```

Smalltalk

- Declaration
```
readFirstChildrenFromCurrentInstance
```
- Use name
```
readFirstChildrenFromCurrentInstance
```

COM

- C++ declaration
```
public void readFirstChildrenFromDetail()
```
- Use name
```
readFirstChildrenFromDetail
```

3.2.2.4. Reading of the Complete Hierarchy of a Current Instance

Operation

This action retrieves all the instances of dependent nodes throughout the Folder, depending on the instance found in the **Instance-presentation** attribute of the root node.

This action is valid if:

- The node's Instance-presentation attribute contains an instance.

If this action is valid, its result is the same as that of a read action on an instance and its complete hierarchy.

This action is always available on all root nodes.

Java

- Declaration
```
public void readAllChildrenFromCurrentInstance() throws
LocalException, ServerException, CommunicationError,
SystemError
```
- Use name
```
Read All Children From Detail
```

Smalltalk

- Declaration
```
readAllChildrenFromCurrentInstance
```
- Use name
```
readAllChildrenFromCurrentInstance
```

COM

- C++ declaration
```
public void readAllChildrenFromDetail()
```
- Use name
```
readAllChildrenFromDetail
```

3.2.2.5. Anticipated Reading of an Instance's Immediate Hierarchy

Operation

This action has the same functionality as a read on an instance and its immediate hierarchy, but allows for an anticipated selection of instances without impacting the GUI.

This action is valid if:

- The instance provided as a parameter does not have an empty value.

If the action is valid:

- The rules are the same as for the read action on an instance and its immediate hierarchy, except that the Instance-presentation attribute of the affected node may contain an empty value.

If the action is not valid:

- The error is added to the Error Object.
- An error event is sent according to the type of error.

This action is always available on root nodes.

Java

- Declaration `public void readFirstChildren({}Data data) throws LocalException, ServerException, CommunicationError, SystemError`
- Use name `Read First Children From {}Data`

Smalltalk

- Declaration `readFirstChildrenFrom: aDataDescription`
- Use name `readFirstChildrenFrom:`

COM

- C++ declaration `public void readWithFirstChildrenFrom({Name of generated class}Data d)`
- Use name `readWithFirstChildrenFrom({Name of generated class}Data d)`

3.2.2.6. Anticipated Reading of an Instance's Complete Hierarchy

Operation

This action has the same functionality as a read on an instance and its complete hierarchy, but it allows for an anticipated selection of instances without impacting the GUI.

This action is valid if:

- The instance provided as a parameter does not have an empty value.

If the action is valid:

- The rules are the same as for the read action on an instance and its complete hierarchy, except that the Instance-presentation attribute of the affected node may contain an empty value.

If the action is not valid:

- The error is added to the Error Object.
- An error event is sent according to the type of error.

This action is always available on root- or dependent-type nodes having at least one dependent node.

Java

- Declaration `public void readAllChildren({}Data data) throws LocalException, ServerException, CommunicationError, SystemError`
- Use name `Read All Children From {Name of generated class}Data`

Smalltalk

- Declaration `readAllChildrenFrom: aDataDescription`
- Use name `readAllChildrenFrom:`

COM

- C++ declaration `public void readWithAllChildrenFrom({Name of generated class}Data d)`
- Use name `readWithAllChildrenFrom({Name of generated class}Data d)`

3.2.3. Management of Paging

3.2.3.1. Reading of the Following Page's Instances

Operation

This action retrieves the next page in a node's collection. When the selected paging mode is of the `extend` type, retrieved instances are compounded with the instances already present in the **Instance-list-presentation** attribute. Locally-created instances which might conflict with retrieved instances have top priority. When the paging is of the `non-extend` type, instances contained in the **Instance-list-presentation** attribute are overridden with the retrieved instances.

This action is valid if:

- The last page of the collection has not been reached yet. Otherwise, this action does not trigger a server access, and sends a “Collection's last-page retrieval” event.
- On a dependent node, a collection must be already defined, or the **Instance-presentation** attribute of the parent node must contain an instance.
- On a root or reference node, if no collection has been defined, this action operates as an instance-selection action.

If the action is valid:

- The Instance-list-presentation attribute is initialized with the result of the query according to the paging-type and to the collection management mode.
- If the paging applies to a root node and is of the `non-extend` type, with a collection management in automatic mode, the Instance-presentation attribute of the node is set to an empty value.
- If the paging applies to a root or dependent node and is of the `extend` type, or with a collection management in automatic mode, its Instance-presentation attribute as well as the Instance- and Instance-list-presentation attributes of the dependent nodes are not modified.
- The “Total number of local instances” counter is initialized.
- The node-selection return message's label and key are initialized for each dependent node if the collection's last instance has been retrieved.
- A no-error-detection event is sent.
- The event “Presence of updatable local instances” is sent, with a collection management in automatic mode, if updatable instances are still present in the local cache.
- The “Collection's fist-page retrieval” event is sent if the action applies to the root or reference node, if the paging type is `non-extend`, with a collection management in automatic mode, and if the page is the first retrieved page in the collection.
- The “Presence of at least one preceding page” event is sent if the action applies to the root or reference node, if the paging type is `non-extend`, with a collection management in automatic mode, and if the page is not the first retrieved page in the collection.
- The “Presence of at least one following page” event is sent if the last instance of the collection has not been retrieved.

- The “Collection's last-page retrieval” event is sent if the last instance of the collection has been retrieved.

If the action is not valid:

- The error is added to the Error Object.
- An error event is sent according to the type of error.

In all circumstances:

- Conversation time counters are set.
- Contextual information attributes –if they are present– are set.

This action is always available on all nodes.

Java

- Declaration `public void readNextPage() throws LocalException, ServerException, CommunicationError, SystemError`
- Use name `Read Next Page`

Smalltalk

- Declaration `readNextPage`
- Use name `readNextPage`

COM

- C++ declaration `public void readNextPage()`
- Use name `readNextPage`

3.2.3.2. Reading of the Preceding Page's Instances

Operation

This action retrieves the previous page of a node's collection. It is designed exclusively for `non-extend`-type paging with a collection management in automatic mode. Instances found in the Instance-list-presentation attribute are systematically overridden by retrieved instances.

This action is valid if:

- The last page of the collection has not been reached yet. Otherwise, this action does not trigger a server access, and sends the “Collection's first-page retrieval” event.
- If no collection is defined, this action operates like an instance-selection action.

If the action is valid:

- The Instance-list-presentation attribute is initialized.
- The Instance-list-presentation attribute is modified.
- The “Total number of local instances” counter is initialized.
- The selection-return message's label and key are initialized if the collection's last instance has been retrieved.
- The Instance- and Instance-list-presentation attributes of dependent nodes are set to empty values.
- A no-error-detection event is sent.
- The “Presence of updatable local instances” event is sent if updatable instances are still present in the local cache.
- The “Presence of at least one preceding page” event is sent if the page is not the first one retrieved in the collection.

- The “Retrieval of a collection's first page” event is sent if the page is the first one in the collection.
- The “Presence of at least one following page” event is sent if the collection's last instance was not retrieved.
- The “Last page retrieval” event is sent if the collection' last instance was retrieved.

If the action is not valid:

- The error is added to the Error Object.
- An error event is sent according to the type of error.

In all circumstances:

- Conversation time counters are set.
- Contextual information attributes –if they are present– are set.

The action is available on root- and reference-type nodes.

Java

- Declaration
```
public void readPreviousPage() throws LocalException,
ServerException, CommunicationError, SystemError
```
- Use name
```
Read Previous Page
```

Smalltalk

- Declaration
```
readPreviousPage
```
- Use name
```
readPreviousPage
```

COM

- C++ declaration
```
public void readPreviousPage()
```
- Use name
```
readPreviousPage
```

3.2.4. Sending of Updates

3.2.4.1. Sending of Local Updates to the Server

Operation

This action sends to the server all the updates performed locally since it was last executed.

Only useful transactions are sent.

This action is valid if:

- At least one local update has been performed.

If the action is valid:

- All updated instances are deleted from the local cache.
- Each modified Logical View instance is updated in the local cache with its last, post-update server image, if the instance-refresh option is set when sending the action.
- Data checking on the server may be activated by setting the appropriate attribute before executing the action.
- A no-error-detection event is sent.

If the action is not valid:

- The error is added to the Error Object.
- An error event is sent according to the type of error.

In all circumstances:

- Conversation time counters are set.
- Contextual information attributes –if they are present– are set.

This action is available on a root node when at least one of the Business Components associated to the Folder's nodes can perform updates.

Java

- Declaration
```
public void updateFolder() throws LocalException,
ServerException, CommunicationError, SystemError
```
- Use name
```
Update Folder
```

Smalltalk

- Declaration
```
updateFolder
```
- Use name
```
updateFolder
```

COM

- C++ declaration
```
public void updateFolder()
```
- Use name
```
updateFolder
```

3.2.4.2. Sending of Updates – Selection

Operation

This action sends to the server all the updates performed locally since it was last executed, and requests a new selection on the root-node instance collection following the processing of the update, if no error is detected by the processing.

Only useful transactions are sent.

This action is valid if:

- At least one local update has been performed.
- Data-refresh on the server is not active.

If the action is valid:

- All updated instances are deleted from the local cache.
- A no-error-detection event is sent.
- All the rules defined for the Selection of a Set of Instances action are activated.

If the action is not valid:

- The error is added to the Error Object.
- An error event is sent according to the type of error.

In all circumstances:

- Conversation time counters are set.
- Contextual information attributes –if they are present– are set.

This action is available on a root node when at least one of the Business Components associated to the Folder's nodes can perform updates.

Java

Not available.

Smalltalk

- Declaration `updateFolderAndSelectInstances`
- Use name `updateFolderAndSelectInstances`

COM

Not available.

3.2.5. Management of Logical Locks

3.2.5.1. Logical Locking of a Current Instance

Operation

This action appropriates a Folder instance for exclusive update. It can apply to a local instance identifier representing an instance which does not yet exist in the Database.

This action is valid if:

- The root node's Selection Criteria attribute contains a Logical View instance identifier.
- The instance is not currently locked.

If the action is valid:

- A no-error-detection event is sent.
- The Folder-lock identifier attribute is initialized with the value returned by the server.
- The Folder changes to the “modifiable” status.

If the action is not valid:

- The error is added to the Error Object.
- An error event is sent according to the type of error.
- The Folder-lock identifier attribute is set to an empty value.
- The Folder changes to the “not-modifiable” status.

In all circumstances:

- Conversation time counters are set.
- Contextual information attributes –if they are present– are set.

This action is available on a root node when the logical lock option is set for this Folder in the VisualAge Pacbase Repository.

Java

- Declaration
```
public void lock() throws LocalException,
ServerException, SystemError, CommunicationError
```
- Use name
```
Lock
```

Smalltalk

- Declaration
```
lock
```
- Use name
```
lock
```

COM

- C++ declaration
```
public void lock()
```
- Use name
```
lock
```

3.2.5.2. Logical Unlocking of a Current Instance

Operation

This action “frees” a Folder instance used for exclusive update when the user chooses not to send locally-updated instances to the server.

This action is valid if:

- The root node's Selection Criteria attribute contains a Logical View instance identifier.
- The instance is locked.

If the action is valid:

- A no-error-detection event is sent.
- The Folder-lock identifier key attribute is set to an empty value.

If the action is not valid:

- The error is added to the Error Object.
- An error event is sent according to the type of error.

In all circumstances:

- Conversation time counters are set.
- Contextual information attributes –if they are present– are set.

This action is available on a root node when the logical lock option is set for this Folder in the VisualAge Pacbase Repository.

Java

- Declaration
```
public void unlock() throws LocalException,
ServerException, SystemError, CommunicationError
```
- Use name
```
Unlock
```

Smalltalk

- Declaration
```
unlock
```
- Use name
```
unlock
```

COM

- C++ declaration
```
public void unlock()
```
- Use name
```
unlock
```

3.2.6. Connection and disconnection to the server

3.2.6.1. Connection to the server

Operation

This action is used to make an explicit connection to the server and enables the user to check the availability of the communication before any transmission of functional query.

Java

Not available.

Smalltalk

- Declaration `connect`
- Use name `connect`

COM

Not available.

3.2.6.2. Disconnection to the server

Operation

This action is used to set free explicitly the resources used by the middleware before leaving an application.

Java

Not available.

Smalltalk

- Declaration `disconnect`
- Use name `disconnect`

COM

Not available.

3.2.7. Management of Dependent Instances

3.2.7.1. Check on the Presence of Dependent Instances

Operation

This action finds out if the Logical View instance contained in the node's Instance-presentation attribute has dependent instances. If this instance was not created locally, and does not contain any dependent instances locally, the system sends this action to the server in order to check for the existence of first-level dependent instances.

This action is valid if:

- The Instance-presentation attribute contains a non-empty value.

If the action is valid:

- A no-error-detection event is sent.
- A “Presence of a dependent instance” or “Absence of dependent instances” event is sent.

If the action is not valid:

- The error is added to the Error Object.
- An error event is sent according to the type of error.

In all circumstances, if the action was passed on to the server:

- Conversation time counters are set.
- Contextual information attributes –if they are present– are set.

This action is available when the node on which it is performed has at least one dependent node.

Java

- Declaration
```
public void checkExistenceOfDependentInstances() throws
LocalException, ServerException, CommunicationError,
SystemError
```
- Use name
```
Check existence of dependent instances
```

Smalltalk

- Declaration
```
checkExistenceOfDependentInstances
```
- Use name
```
checkExistenceOfDependentInstances
```

COM

- C++ declaration
```
public void checkExistenceOfDependencies()
```
- Use name
```
checkExistenceOfDependencies
```

3.2.8. Management of User Services

3.2.8.1. Execution of User Services

Operation

This action executes a User Service associated to a node and to its dependent nodes for which a “User Service to be executed” is set.

This action operates if:

- At least one of the relevant nodes contains a non-empty value in the “User Service to be executed” attribute.

If the action is valid:

- A no-error-detection event is sent.
- The presentation attribute of instances returned by a User Service is initialized.
- The “Number of Logical View instances processed by a User Service” attribute is recalculated.

If the action is not valid:

- The error is added to the Error Object.
- An error event is sent according to the type of error.

In all circumstances:

- Conversation time counters are set.
- Contextual information attributes –if they are present– are set.

This action is available if the Business Component associated to the node contains at least one User Service.

Java

- Declaration
```
public void executeUserService() throws ServerException, CommunicationError, SystemError, LocalException
```
- Use name
```
Execute User Service
```

Smalltalk

- Declaration
```
executeUserService
```
- Use name
```
executeUserService
```

COM

- Declaration
```
public void executeUserService()
```
- Use name
```
executeUserService
```

3.2.9. Management of Asynchronous Conversations

3.2.9.1. Deferred Retrieval of a Reply

Operation

This action retrieves the reply associated to a query sent with the asynchronous communication type.

This action is valid if:

- The communications protocol used to send the query supports asynchronous conversations.
- The conversation type is asynchronous.
- The identifier of the query specified as parameter is valid and known.

If this action is valid and the query available:

- The rules applied are the same as those governing the action which sent the query –if the query was executed in synchronous mode.
- The “Number of pending replies” attribute is decremented by 1.
- Contextual information attributes –if they are present– are set.

If the action is not valid:

- The error is added to the Error Object.
- An error event is sent according to the type of error.

If the query is not available:

- The “Unavailable-reply retrieval” event is sent.

In all circumstances:

- Conversation time counters are set.

This action is always available on a root node.

Java

- Declaration

```
public Boolean
getReply(com.ibm.vap.generic.ServerActionContext aContext)
throws LocalException, ServerException, CommunicationError,
SystemError
```

- Use name

```
Get Reply
```

Smalltalk

- Declaration

```
getReplyOf: aReplyId
```

- Use name

```
getReplyOf:
```

COM

- C++ declaration

```
public Boolean getReply(ServerActionContext s)
```

- Use name

```
getReply(ServerActionContext s)
```

3.2.9.2. Deletion of a Retrieval-Pending Reply

Operation

This action explicitly deletes a reply identifier associated to a query.

This action is valid if:

- The communications protocol used to send the query supports asynchronous conversations.
- The conversation type is asynchronous.
- The identifier of the query specified as parameter is valid and known.

If this action is valid and the query available:

- The reply identifier key is deleted and the reply is purged from the Message Manager.
- The “Number of pending replies” attribute is decremented by 1.

If the action is not valid:

- The error is added to the Error Object.
- An error event is sent according to the type of error.

If the query is not available:

- The “Unavailable-reply retrieval” event is sent.

In all circumstances:

- Conversation time counters are set.

This action is always available on a root node.

Java

Not available.

Smalltalk

- Declaration `resetPendingReplyOf: aReplyId`
- Use name `resetPendingReplyOf:`

COM

Not available.

3.2.9.3. Check on a Message Identifier's Validity

Operation

This action finds out if a query identifier is valid and known.

If the action is valid:

- The action returns `true` if the key is valid, or `false` otherwise.

If the action is not valid:

- The error is added to the Error Object.
- An error event is sent according to the type of error.

If the query is not available:

- The “Unavailable-reply retrieval” event is sent.

This action is always available on a root node.

Java

- Declaration `public Boolean isReplyValid(com.ibm.vap.generic.ServerActionContext aContext)`
- Use name `Checks the validity of the request reply`

Smalltalk

- Declaration `isReplyIdValid: aReplyId`
- Use name `isReplyIdValid:`

COM

- C++ declaration `public Boolean isReplyValid(ServerActionContext s)`
- Use name `isReplyValid(ServerActionContext s)`

3.2.10. Management of Executed Actions

3.2.10.1. Re-Execution of the Last Action

Operation

This action re-executes the latest action executed locally or on the server.

Validity an return-processing rules for this action are those of the action being re-executed.

Java

Not available.

Smalltalk

- Declaration `redoLastAction`
- Use name `redoLastAction`

COM

Not available.

3.2.11. Sub-Schema Management

Operation

This action retrieves, by calling the Business Component associated with the Logical View, the values of the Data Elements which do not belong to the sub-schema selected via the **subSchema** attribute.

Upon the correct return of this action, the instance is considered to be complete, so its associated implicit sub-schema is reset. Any subsequent modification is then performed with no associated sub-schema.

Before this action is executed, the Data Elements which belong to the sub-schema may have been modified locally.

This action is available if the Business Components manage the presence of Data Elements (**VECTPRES=YES** or **CHECKSER=YES**).

Java

- Declaration
```
public completeInstance throws LocalException,
ServerException, CommunicationError, SystemError
```
- Use name
```
Complete Instance
```

Smalltalk

- Declaration
```
completeInstance
```
- Use name
```
completeInstance
```

COM

- C++ declaration
```
public completeInstance()
```
- Use name
```
completeInstance
```

4. Events

4.1. Management of Paging

For COM, all the events described in the chapter are to be retrieved in the events stack associated to each Proxy through the `public String popServerEvent()` method, as long as `public Int getServerEventsCount()` does not return zero. Retrieved Strings correspond to the codes of the described events.

4.1.1. Signal of Retrieval of a Collection's Last Page

Sending rules

This signal is sent by a node when an instance-set selection action or a paging action returns a page containing the last instance of the selected collection. This event is available for root and reference nodes, and for dependent nodes with a maximum cardinal value of 1.

Java

- Code `noPageAfter`

Smalltalk

- Code `noPageAfter`

COM

- String code `NO_PAGE_AFTER`

4.1.2. Signal of Retrieval of a Collection's First Page

Sending rules

This signal is sent by a node when an instance-set selection action or a paging action returns a page containing the first instance of the selected collection. This event is available for root and reference nodes, when the paging mode is of the `non-extend` type with a collection management in automatic mode.

Java

- Code `noPageBefore`

Smalltalk

- Code `noPageBefore`

COM

- String code `NO_PAGE_BEFORE`

4.1.3. Signal of Presence of at Least One Following Page

Sending rules

This signal is sent by a node when an instance-set selection action or a paging action returns a page which does not contain the last instance of the selected collection. This event is available for root and reference nodes, and for dependent nodes with a maximum cardinal value of 1.

Java

- Code `pageAfter`

Smalltalk

- Code `pageAfter`

COM

- String code `PAGE_AFTER`

4.1.4. Signal of Presence of at Least One Preceding Page

Sending rules

This signal is sent by a node when an instance-set selection action or a paging action returns a page which does not contain the first instance of the selected collection. This event is available for root and reference nodes, when the paging mode is of the `non-extend` type with a collection management in automatic mode.

Java

- Code `pageBefore`

Smalltalk

- Code `pageBefore`

COM

- String code `PAGE_BEFORE`

4.2. Management of Unit Reads

4.2.1. Signal of Reading of a Record not Found

Sending rules

This signal is sent by a node when an instance-reading action has not returned the required instance.
This action is always available for all nodes.

Java

- Code `notFound`

Smalltalk

- Code `notFound`

COM

- String code `NOT_FOUND`

4.3. Management of Simultaneous Selections

4.3.1. Signal of Non-Participation to a Simultaneous Read

Sending rules

This signal is sent by a node following a retrieval action on the last event associated to the last server selection, when the node did not participate in this selection action.
This action is always available for all nodes.

Java

- Code `notRead`

Smalltalk

- Code `notRead`

COM

- String code `NOT_READ`

4.4. Management of Logical Locks

4.4.1. Signal of Assigned Logical Lock

Sending rules

This signal is sent by a root node following an action of logical-lock request on a valid instance.
This action is available for a root node when the logical-lock option is coded for this node in the VisualAge Pacbase Repository.

Java

- Code `lockSuccessful`

Smalltalk

- Code `lockSuccessful`

COM

- String code `LOCK_SUCCESSFUL`

4.4.2. Signal of Unsuccessful Logical Lock

Sending rules

This signal is sent by a root node following an action of logical-lock request on a instance when this instance is already set to exclusive update mode by another user.

This action is available for a root node when the logical-lock option is coded for this node in the VisualAge Pacbase Repository.

Java

- Code `lockFailed`

Smalltalk

- Code `lockFailed`

COM

- String code `LOCK_FAILED`

4.5. Management of Dependent Instances

4.5.1. Signal of Presence of at Least One Dependent Instance

Sending rules

This signal is sent by a node following a check action on the presence of dependent instances, when the instance concerned has at least one dependent instance. This action is always available for root- and dependent-type nodes.

Java

- Code `dependentInstances`

Smalltalk

- Code `dependentInstances`

COM

- String code `DEPENDENT_INSTANCES`

4.5.2. Signal of Absence of Dependent Instances

Sending rules

This signal is sent by a node following a check action on the presence of dependent instances when the instance concerned has no dependent instances. This action is always available for root- and dependent-type nodes.

Java

- Code `noDependentInstances`

Smalltalk

- Code `noDependentInstances`

COM

- String code `NO_DEPENDENT_INSTANCES`

4.6. Management of Asynchronous Conversations

4.6.1. Signal of Execution of an Asynchronous Server Action

Sending rules

This signal is sent by a root node when the query processed by the server uses the asynchronous conversation type. This action is always available for a root node.

Java

Replaced with the `AsynchronousRequestException` exception.

Smalltalk

- Code `asyncRequest`

COM

Not available.

4.6.2. Signal of Unavailable-Reply Retrieval

Sending rules

This signal is sent by a root node when the Message Manager cannot provide the reply associated to the query's reply identifier.

This action is always available for a root node.

Java

Replaced with the `false` value, returned by the `getReply` method.

Smalltalk

- Code `replyPending`

COM

Replaced with the `false` value, returned by the `getReply` method.

4.7. Management of a Logical View's Instance Collection

4.7.1. Signal Preceding an Instance Collection Change

Sending rules

This signal is sent when at least one local update has been performed on the current instance collection and a new instance-selection or paging action is performed.

It guarantees that the server update is performed.

Java

- Code `aboutToChangeSelection`

Smalltalk

- Code `aboutToChangeSelection`

COM

- Code `ABOUT_TO_CHANGE_SELECTION`

4.8. Management of errors

4.8.1. Signal of Restoration of the Error Context

Sending rules

This signal is sent by a node when a required restoration of an error-linked context performed correctly, i.e. the Instance-presentation attribute or the **Presentation of an instance linked to a User Service** attribute is initialized with the instance that caused the error.

This action is always available for a root- or dependent-type node.

Java

Not available.

Smalltalk

- Code `errorContextRestored`

COM

Not available.

5. Data Elements Handling Public Interface

5.1. Management of a Data Element's Contents

Description

This attribute displays the Data Element's contents.

This attribute is always available for Data Elements defined in the `DataDescription`, `SelectionCriteria`, and `UserContext` classes.

Java

- Type — Depends on the type of Data Element (`java.lang.String`, `int`, `long`, `double`, or `java.util.Date`)
- Internal code — `<DataElementCode>`
- Use name — `<Data Element Clear Name>`
- get/set —
```
public [type] get<DataElementCode> ()
public void set<DataElementCode> (T-type)
```

Smalltalk

On top of the implicit management via the *QuickForm*, this attribute may be managed explicitly, as follows:

- Type — `VpcsString`, `VpcsInteger`, `VpcsDecimal`, `VpcsDate`, or `VpcsTime`
- Internal code — `<DataElementCode>`
- Use name — `<DataElementCode>`
- get/set — `<DataElementCode>` / `<DataElementCode>:`

COM

- Type — Depends on the type of Data Element
- Internal code — `<DataElementCode>`
- Use name — `<DataElementCode>`
- get/set C++ —
```
public [Type] get<DataElementCode> ()
public void set<DataElementCode> (Type t )
```

5.2. Management of Authorized-Value Codes

Description

This attribute provides the authorized values associated to a Data Element.

This attribute is always available for Data Elements containing authorized values and defined in the **`DataDescription`** class.

☞ For the VisualAge Smalltalk environment, Data Elements are handled implicitly via the *QuickForm*.

Java

- Type: `[ Type [ ] ]`
- Internal code: `<DataElementCode>Values`
- Use name: `<DataElementCode> Values`
- get/set: `public [Type[ ] ] get<DataElementCode>Values ( )`
  set not available

COM

Available with a browsing API for collection-type attributes.

- Nb of elements: `public Int <DataElementCode>ValidValuesCount()`
- Element: `public Type <DataElementCode>ValidValuesAt(Int i)`

5.3. Management of Authorized Values

Description

This attribute displays the labels of authorized values associated to a Data Element.

This attribute is always available for Data Elements which contain authorized values and which are defined in the **`DataDescription`** class.

☞ For the VisualAge Smalltalk environment, Data Elements are handled implicitly via the *QuickForm*.

Java

- Type: `String [ ]`
- Internal code: `<DataElementCode>Labels`
- Use name: `<DataElementCode> Labels`
- get/set: `public String[ ] get<DataElementCode>Labels ( )`
  set not available

COM

Available with a browsing API for collection-type attributes.

- Nb of elements: `public Int <DataElementCode>ValidLabelsCount()`
- Element: `public Type <DataElementCode>ValidLabelsAt(Int i)`

5.4. Management of the Validity of a Data Element's Contents

Description

This action specifies whether the contents of a Data Element are valid or not.

This action is always available for Data Elements which contain authorized values and which are defined in the **DataDescription** class.

☞ For the VisualAge Smalltalk environment, Data Elements are handled implicitly via the *QuickForm*.

Java

- Declaration **public Boolean is<DataElementCode>Valid ()**
- Use name **Is<DataElementCode> Valid**

This method has been **deprecated** since version 2.5v07, it is replaced by the **get<DataElementCode>Error** method.

- Declaration **public DataFieldError get<CodeRubrique>Error ()**
- Use name **<DataElementCode> Error**

This method returns a **DataFieldError** instance which indicates the nature of the error detected on the field or the **null** value if the Data Element contents are empty.

COM

Not available.

5.5. Management of a Data Element's Presence

Description

These actions specify whether the Data Element is absent (empty contents) or present (contents not empty).

The first action is always available for Data Elements defined in the **DataDescription** and **UserDataDescription** classes

- of all the nodes for Java and COM
- of the nodes whose Business Component includes the **NULLMNGT=YES** option for Smalltalk.

The second action is always available for Data Elements defined in the **DataDescription** and **UserDataDescription** classes

- of all the nodes for Java and COM
- of the nodes whose Business Component includes the **NULLMNGT=YES** option and a user service for Smalltalk.

Before this action is executed, all Data Elements are considered to be:

- present for Smalltalk
- absent for Java and COM, except if a default value has been specified in VisualAge Pacbase.

Java

- Declaration
```
public Boolean is<DataElementCode>Present ()
public void set<DataElementCode>Present (Boolean b)
```
- Use name
```
<DataElementCode> Present
```

Smalltalk

- Declaration
```
isNull:<DataElementCode>
setNull: (Boolean b) on: <DataElementCode>
```
- Use name
```
isNull:
setNull: on:
```

COM

- C++ declaration
```
public Boolean is<DataElementCode>Present ()
public Boolean set<DataElementCode>Present (Boolean b)
```
- Use name
```
is<DataElementCode>Present
set<DataElementCode>Present(Boolean b)
```

5.6. Management of a Data Element's Check

Operation

These actions specify whether the Data Element is to be checked or not.

These actions are always available for Data Elements defined in the `DataDescription` and `UserDataDescription` classes of the root or dependent nodes whose Business Component includes the `NULLMNGT=YES` and `CHECKSER=YES` options and an update service.

Before this action is executed, all Data Elements are to be checked (if the `serverCheckOption` attribute is set to true).

Java

- Declaration
```
public setCheck(int index, boolean aBoolean)
```
- Use name
```
Check Flag for the Index’s field
```

The index of the Data Element to be checked is found via the following method:

- Declaration
```
public int get<DataElementCode>Index()
```
- Use name
```
<DataElementCode> Index
```

Smalltalk

- Declaration
```
setCheck: aBoolean on: <DataElementCode>
```
- Use name
```
setCheck: on:
```

COM

- C++ declaration
```
public setCheck(Int fieldIndex, Boolean b)
```
- Use name
get not available / `setCheck(fieldIndex,b)`

6. Error Manager Public Interface [Smalltalk, COM]

This interface contains the attributes, actions, and events used to manage all the local, communications and server errors, for Smalltalk and COM.

For Smalltalk, the Error Manager corresponds to the `errorManager` attribute.

For COM, errors are represented in the `VAPERROR` error manager.

6.1. Attributes

6.1.1. List of Errors

Description

This attribute contains the instances corresponding to errors returned by a local action or by a server action. In particular, each instance contains the following attributes:

- Error type
- Action on which the error occurred
- Complete error key
- Error label
- Error gravity

This attribute is available for read-only access.

Smalltalk	
• Type	`VpcsOrderedCollection` containing `VpcsError`
• Internal code	`errorList`
• Use name	`errorList`
• get/set	`errorList` / set not available

COM	
• C++ declaration	`public VapError getErrorsElementAt(Int i)`
• Use name	`getErrorsElementAt(Int i)`

6.1.2. Error Type

Description

This COM attribute contains the error type.

- Local
- Server
- Communication

This attribute is available for read-only access.

COM

- Type `String`
- Internal code `getType`
- Use name `getType`
- get/set `public String getType()`/ set not available

6.1.3. Error-Producing Action

Description

This COM attribute contains the action which caused the error.

This attribute is available for read-only access.

COM

- Type `String`
- Internal code `getAction`
- Use name `getAction`
- get/set `public String getAction ()`/ set not available

6.1.4. Error Key

Description

This COM attribute contains the error key (see the *Pacbench C/S User's Guide, Vol. III: Graphic Clients* for the list of errors).

This attribute is available for read-only access.

COM

- Type `String`
- Internal code `getKey`
- Use name `getKey`
- get/set `public String getKey ()`/ set not available

6.1.5. Error Label

Description

This COM attribute contains the error label.

This attribute is available for read-only access.

COM

- Type `String`
- Internal code `getLabel`
- Use name `getLabel`
- get/set `public String getLabel ()`/ set not available

6.1.6. Error Gravity

Description

This COM attribute contains the error gravity.

- Exception
- Error

This attribute is available for read-only access.

COM

- Type `String`
- Internal code `getGravity`
- Use name `getGravity`
- get/set `public String getGravity ()`/ set not available

6.1.7. Number of Errors

Description

This attribute contains the number of errors returned by a local action or by a server action.

It is available for read-only access.

Smalltalk

- Type `Integer`
- Internal code `errorCount`
- Use name `errorCount`
- get/set `errorCount` / set not available

COM

- C++ declaration `public Int getErrorsCount()`
- Use name `getErrorsCount`

6.2. Actions

6.2.1. Context Restoration

Operation

From an error message key, this action sets the **Instance-presentation** attribute or the **Presentation of an instance linked to a User Service** attribute with the Logical View instance which caused the error, if the context related to the error is to be restored.

This action is valid if:

- The complete error key used as parameter is valid.

If the action is valid:

- The “No context retrieval” attribute is set to an empty value or to the standard error message if the context related to the error cannot be restored.
- If the context related to the error is to be restored, the presentation attribute of the Logical View instance or User Service-linked instance affected by the error is set.
- The Instance- and Instance-list-presentation attributes of the dependent nodes are set according to the Global setting rule for these attributes.
- The context-restoration signal associated to the error is sent.

This action is always available.

Smalltalk

- Declaration
```
restoreContextOfSelectedError: anError
```
- Use name
```
restoreContextOfSelectedError:
```

COM

Not available.

6.2.2. Communication Management: Attribute Assignment based on its Name

Description

This attribute makes it possible to assign a value to the attribute passed as a parameter. It enables to define the communication dll in use (`MwAdaper`, `GwAdapter`,etc..) as well as the required attributes (location, port, host...) :

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

This attribute is always available on the root node.

COM

- Type `Variant`
- Code interne `property`
- Nom d'utilisation `getProperty`
- get/set

```
public Variant getProperty (String attribute_name)
public void setProperty(String attribute_name,Variant value)
```

6.3. Events

For COM, all the events described in the chapter are to be retrieved in the events associated to each Proxy, through the `public String popServerEvent()` method as long as `public Int getServerEventsCount()` does not return zero. Retrieved Strings correspond to the codes of the described events.

6.3.1. Error Management

6.3.1.1. Signal of No Error Detection

Sending rules

This signal is sent by the Error Manager when no local, server or system error is detected.

This event is always available.

Smalltalk

- Code `noError`

COM

- Codes `NO_ERROR`

6.3.1.2. Signal of Local Error Retrieval

Sending rules

This signal is sent by the Error Manager when a local action detects an error.

This event is always available.

Smalltalk

- Code `localError`

COM

- Code `LOCAL_ERROR`

6.3.1.3. Signal of Server Error Retrieval

Sending rules

This signal is sent by the Error Manager when a server detects a logical access error, a user error or a Logical View data check error.

This event is always available.

Smalltalk

- Code `serverError`

COM

- Code `SERVER_ERROR`

6.3.1.4. Signal of System Error Retrieval

Sending rules

This signal is sent by the Error Manager when a server detects a serious error, such as a release or version discrepancy between the client component and the associated Business Component.

This event is always available.

Smalltalk

- Code `systemError`

COM

- Code `SYSTEM_ERROR`

6.3.1.5. Signal of Communications Error Retrieval

Sending rules

This signal is sent by the Error Manager when a communication problem is detected during an exchange between a client and a server.

This event is always available.

Smalltalk

- Code `fatalError`

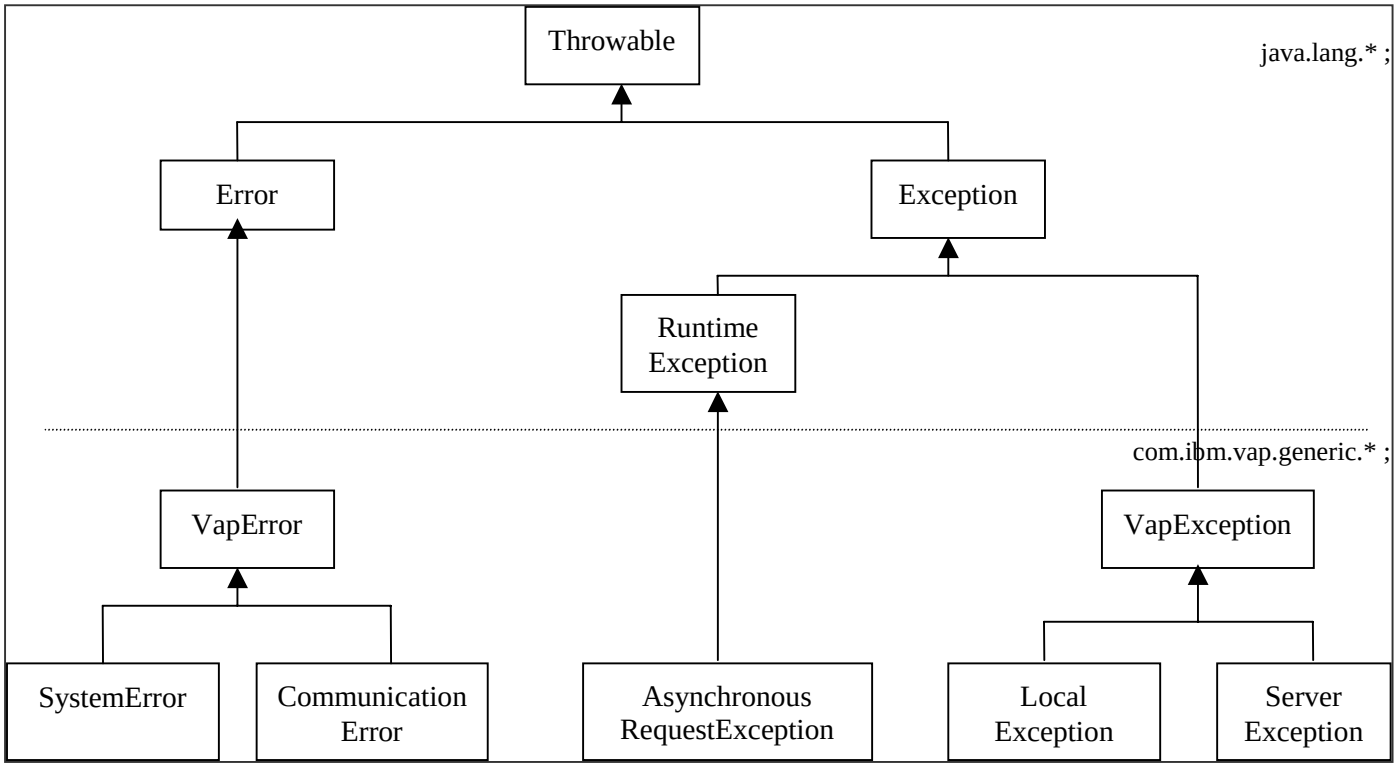
COM

- Code `FATAL_ERROR`

7. Management of Errors for Java Target

The management of the errors associated to the handling of Java Proxies is based on the raise of exceptions mechanism.

Four classes enable to carry errors or exceptions issued from a VisualAge Pacbase Proxy. The four classes are available to the developper and they inherit from **`java.lang.Throwable`** in the following way:



The **`com.ibm.vap.generic.AsynchronousRequestException`** exception is raised at a server acces whenever the Proxy is in asynchronous mode.

The execution of some methods of the Proxy require the control of any exception inheriting from **`com.ibm.vap.generic.VapException`** (refer to the declarations documented in the manual to know the different types of exceptions that can be sent by a method).

It is also highly recommended to control errors inheriting from **`com.ibm.vap.generic.VapError`** although the control is optional for this type of class in Java.

Note: An excessive use of some Proxy methods may also raise the **`java.lang.IllegalStateException`** exception, like in the case of the use of a method which does not match with the Pacbase Proxy definition (example: using the **`createUserInstance`** method although no user service was defined in the server Pacbase description).

7.1. Classes Related to the Management of Errors

7.1.1. Communication Errors

Name of the exception

`com.ibm.vap.generic.CommunicationError`

Note

The exception is raised if an error occurs in the communication string with the server.
The message carried by each instance of the class gives information concerning the detected error (the `getMessage()` method of the class).

7.1.2. System Errors

Name of the exception

`com.ibm.vap.generic.SystemError`

Note

The exception is raised by system errors.

This type of error represents an internal and irretrievable error. It can be detected either by the client or by the server.

If the server detects system errors, messages associated to these errors are represented by instances of `com.ibm.vap.generic.ServerMessage`. They are available through the `java.util.Enumeration serverMessages()` method from the `com.ibm.vap.generic.SystemError` class.

☞ Refer to the *User's Guide Volume III – Graphic Clients* for the list of the system errors.

7.1.3. Local Errors

Name of the exception

`com.ibm.vap.generic.LocalException`

Note

The exception is raised by local errors. These errors are detected by the client.

The exception holds a property of int type (`int getLocalExceptionKey()`) that enables the error identification that lead to the raise of the exception (wrong creation, invalid instance, …).

☞ Errors responsible for the exception are described in the *User's Guide Volume III – Graphic Clients* and also in the HTML documentation associated with the generic classes: Package `com.ibm.vap.generic.`

7.1.4. Server Errors

Name of the exception

`com.ibm.vap.generic.ServerException`

Note

The exception is raised by server errors. It is raised upon the reception of logical error message(s) detected by the server. These logical error messages are represented by instances of `com.ibm.vap.generic.ServerMessage`. The list of error messages received from the server is available *via* the `java.util.Enumeration serverMessages()` method from the `com.ibm.vap.generic.ServerException` class.

In the case of errors detected at the update service, it is possible to restore the context of the Proxy related to the update request (restoration of the selection tree and of the detail) for each error.

The class `com.ibm.vap.generic.ServerException` indicates if there are restorable errors in the list of errors detected by the method (`boolean isContextRestorable()`).

7.1.4.1. Error Message Received from the Server

Error messages received from the server are represented by objects with the `com.ibm.vap.generic.ServerMessage` type.

This interface offers methods enabling also to fetch the error key (`String key()`), the error message label (`String label`()) and the error message label formatted by the Client (`String localLabel()`).

For the description of the local formatting principle for error labels, refer to the 7.2 section, *Customizing error labels*, on page 127.

7.1.4.2. Error Message Received from the Server on the Update

This interface inherits from `com.ibm.vap.generic.ServerMessage` interface.

Objects of this type represent error messages received from the server; these errors have been detected at the execution of update services.

This interface offers methods allowing to know and to restore the context of the Proxy related to the update request:

- `boolean isContextRestorable`(): indicates if the error context is « restorable »,
- `void restoreContext() throws LocalException`: triggers the restoration of the context related to the error update,
- `DataDescription erroneousData`(): returns the `DataDescription` class which update has failed,
- `HierarchicalProxyLv erroneousProxy()`: returns the `HierarchicalProxy` class that handles the update request that has failed.

7.2. Customizing Error Labels

Customizing labels for errors associated with the handling of Proxies is possible with Java Proxies. The customization is based on the use of internationalization and dynamic formatting technics of labels provided by Java language: the differents labels are stocked in a ressources file (**vaperror.properties**) that will be charged according to the geographical context (the context beeing provided by the default **Locale**). The file is structured on a key-value relation mode where the value corresponds to a label.

Also, each label must respect the formality of the « *patterns* » used by Java when handling labels with variable parts (**java.text.MessageFormat**).

The error labels stored in the the file are:

- the message associated with an instance of **com.ibm.vap.generic.LocalException**, **SystemError** and **ServerException** (available through method **String getMessage()**),
- the label of objects with **com.ibm.vap.generic.ServerMessage** type locally formatted (**String localLabel()**).

7.2.1. Label of Local Errors

Local errors are instances of **LocalException**.

Keys that enable to find local error labels correspond to the name of the constants defined on the **com.ibm.vap.generic.LocalException** class. The keys represent the different types of local errors and are prefixed with **LOCAL_**. Doubles with error codes defined by the user in the error label file are thus avoided.

Example : *the key allowing to find the label associated to a local exception with INVALID_INSTANCE type, will be LOCAL_INVALID_INSTANCE.*

7.2.2. Local Labels of Error Messages Received from the Server Component

Error messages received from the server are set up with two information: a key and a label.

The key of the error message is going to be read in order to identify the key of storage of the local label associated to the error.

The structure of the error message key is described in the *User's Guide Volume III – Graphic Clients.*

In the case of a system error, the access key to the local label corresponds to the **SYSTEM_** prefix followed by characters comprised between columns 14 and 19 if they have a significant value (not blank) or followed by characters comprised between columns 10 and 13, if the value is not significant.

In the case of error servers (user errors, for instance), the access key to the local label corresponds to the characters comprised between columns 14 and 19 if they have a significant value and if the character in column 20 is blank. If the character of column 20 is 2 or 5, the error key is respectively **REQUIRED** and **VALUE**. If the characters comprised between columns 14 and 19 have no significant value, the access key to the local label corresponds to the character comprised between columns 22 and 25.

7.2.3. Labels for Server and System Errors

The label for **com.ibm.vap.generic.ServerException** exceptions type and **com.ibm.vap.generic.SystemError** errors are respectively available with the **VAP_SERVER_EXCEPTION** and **VAP_SYSTEM_ERROR** keys.

7.2.4. Example of Errors Label File

```
# This file defines the default error labels in VisualAgePacbase for Java.
# The labels stored in the bundle are protential java.text.MessageFormat
pattern.
# The possible arguments in the label are :
# {0} = library
# {1} = server name
# {2} = view code
# {3} = data id
# {4} = attribute name
# {5} = attribute value
# {6} = technical label (technical message from the local cache or message
label from the server)
# Note : Those arguments are filled within the error context. They may not
have value
# if the argument is not meaningful in the error context.

# Local exception error
LOCAL_PARENT_INSTANCE_MISSING = Parent instance missing (data id: {3})
LOCAL_CURRENT_INSTANCE_MISSING = Current instance missing
LOCAL_SERVER_UPDATE_REQUIRED = Server update required (data id: {3})
LOCAL_UNKNOWN_INSTANCE = Unknown instance (data id: {3})
LOCAL_INVALID_INSTANCE = Invalid instance
LOCAL_INSTANCE_NOT_LOCKED = Instance not locked (data id: {3})
LOCAL_INVALID_CREATION = Invalid creation (data id: {3})
LOCAL_INVALID_CHANGE = Invalid change (data id: {3})
LOCAL_INVALID_DELETION = Invalid delete (data id: {3})
LOCAL_INVALID_INIALIZATION = Invalid initialization (data id: {3})
LOCAL_CARDINALITY_VIOLATION = Cardinality violation {6} (data id: {3})
LOCAL_INSTANCE_ALREADY_LOCKED = Already locked instance (data id: {3})
LOCAL_CURRENT_USER_INSTANCE_MISSING = Current user instance missing
LOCAL_REFERING_INSTANCE_MISSING = Referring instance missing
LOCAL_ASYNCHRONOUS_VIOLATION = Asynchronous violation ({6})
LOCAL_UNKNOWN_CONTEXT = Unknown context
LOCAL_VALUE_REQUIRED = Required item: {4} (data id: {3})
LOCAL_VALUE_ERROR = Value error: {4} (value: {5}, data id: {3})
LOCAL_LENGTH_ERROR = Length error on instance field: {4} (value: {5}, data
id: {3})
LOCAL_SUBSCHEMA_ERROR = Field {4} is out of sub-schema (value: {5}, data
id: {3})
LOCAL_FOLDER_USER_CONTEXT_LENGTH_ERROR = Length error on instance field in
folder user context: {4} (value: {5})
LOCAL_REFERENCE_USER_CONTEXT_LENGTH_ERROR = Length error on instance field
in a reference user context: {4} (value: {5})

# Server Service Error Messages
REQUIRED = Required value: {4} (library: {0}, server: {1}, view: {2})
VALUE = Value error: {4} (library: {0}, server: {1}, view: {2})
DUPL = Invalid creation (library: {0}, server: {1}, view: {2})
NFND = Invalid delete or modification (library: {0}, server: {1}, view:
{2})
LOCKED = Already locked instance (library: {0}, server: {1})
NTLOCK = Instance not locked (library: {0}, server: {1})

# System Error Messages
SYSTEM_STRU = Structure error onto logical view (library: {0}, server: {1})
SYSTEM_VERS = Version error (library: {0}, server: {1})
```

```
SYSTEM_VIEW = Unknown view (library: {0}, server: {1})
SYSTEM_SERV = Unknown service (library: {0}, server: {1})
SYSTEM_LTH = Length view error (library: {0}, server: {1})
SYSTEM_LSRV = Length received message error (library: {0}, server: {1})
SYSTEM_NUVE = Version error in business component (library: {0}, server:
{1})
SYSTEM_PCVLTH = Message length error (library: {0}, server: {1})
SYSTEM_MISPCV = Components out of phase
SYSTEM_ACCESS = Data access error {6} (library: {0}, server: {1}, view:
{2})
SYSTEM_LKABSC = Invalid absence of lock processing (library: {0}, server:
{1})
SYSTEM_WF00 = Temporary file access error or Database connect error (error
: {6})
SYSTEM_TAND = Pathsend error {6} (library: {0}, server: {1})
SYSTEM_PILO = Pilot Record not found during user buffer processing
(library: {0}, server: {1})
SYSTEM_EXT1 = Extract method : PCV syntax error or size error (library:
{0}, server: {1})
SYSTEM_EXT2 = Unknown extract method (library: {0}, server: {1})
SYSTEM_USR1 = User service not found (library: {0}, server: {1})
SYSTEM_USR2 = Size error for user service (library: {0}, server: {1})
SYSTEM_USR3 = Unknown user service (library: {0}, server: {1})

# Internal Exception Labels
VAP_SERVER_EXCEPTION = A Server Exception occurred.
VAP_SYSTEM_ERROR = A System Error occurred
```

8. INDEX

Classes

COM actions

COM attributes

COM events

Java events

Java methods

Java properties

Smalltalk actions

Smalltalk attributes

Smalltalk events