VisualAge Pacbase

IBM

Pacbench C/S Applications Business Logic

Version 3.5

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/awdtools/vapacbase/productinfo.htm

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

First Edition (February 2004)

This edition applies to the following licensed program:
VisualAge Pacbase Version 3.5
Comments on publications (including document reference number) should be sent electronically through the Support Center Web site at:
http://www.ibm.com/software/awdtools/vapacbase/support.htm
or to the following postal address:
IBM Paris Laboratory
VisualAge Pacbase Support
1 place J.B. Clément
93881 Noisy-le-Grand Cedex France

Table of Contents

A detailed *Table of Contents* is presented in the following pages

Table of Contents

Chapter 5: Communications Monitor (graphic applications) 117

Chapter 6: Folder and Folder Views (graphic applications) 123

Chapter 7: Generation and Extraction.... 139

Chapter 8: Appendix : Summary of the Generated Business Component............ 141

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 contac :IBM Paris Laboratory, SMC Department,1 place J.B. Clément, 93881 Noisy-le-Grand Cedex, 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

Foreword

The objective of this volume is to lead you through the process of developing Server components, using the Business Logic function of the Pacbench/CS module. The specification of such Server components is done with the Pacbench module in the **VisualAge Pacbase WorkStation**.

☞ Starting with the 3.0 Version of VisualAge Pacbase, **Developer workbench** offers a new **eBusiness module**, using a different set of entities than that of Pacbench/CS. Although both models coexist in the current VisualAge Pacbase Databases, you may sometimes find mentions of the 2.5 and 3.0 Metamodels, referring respectively to the Pacbench/CS entities and to the eBusiness entities. Please note that the development of Server components with the eBusiness module is documented in the Developer workbench online help.

The end of this book includes an Index in which you can find generation options, areas of the generated code and the main concepts that are dealt with in this manual.

Prerequisites

Before reading this volume, you must be familiar with the major principles used in the development of Client / Server applications with VisualAge Pacbase. If not, refer to the *Developer's Documentation / eBusiness and Pacbench/CS Applications:* ***Concepts & Architectures***.

The *Developer's Documentation / eBusiness and Pacbench/CS Applications:* ***Graphic Presentation*** contains all the information useful for integrating Server components and Java or COM client applications.

Using the Pacbench/CS **Business Logic** implies a working knowledge of a number of VisualAge Pacbase components, documented in the following manuals:

- *Data Dictionary* *,
- *Structured Code* *,
- *VisualAge Pacbase WorkStation User Interface Guide*,
- *Pacbench/CS Business Logic & TUI Clients* *,
- in case of data storing in a DBMS, see the
  *Database Description Manual**, specific to the DBMS in use.

* The contents of these manuals are available as online help in the VisualAge Pacbase Workstation.

Typographical conventions in use

The `courier` font is used for any character set to be entered, displayed or corresponding to generated code.

Italics is used for titles of publications and chapters in cross-references.

The following icons are used:

☞ note, remark, important point

👓 cross-reference to another location in the documentation

💡 hint or useful piece of information

🗔 action to be performed using a Tool or an Editor

⚠ Precaution to be taken (for risky or irreversible action…)

Terminological conventions in use

- The following Pacbench/CS entities are involved in the development of a Server component:

FUNCTIONAL NAME	ENTITY
Business Component	Business Component (**O** entity, **S**-type)
Error Message Server	Business Component (**O** entity, **E**-type)
Folder	Folder (**O** entity, **F**-type)
Folder View	Folder View (**O** entity, **FV**-type)
Initialization/Termination Component	Business Component (**O** entity, **IT**-type)
Communications Monitor	Business Component (**O** entity, **FM**-type)

- A `FVP` refers to a Folder View Proxy.

Chapter 1: Business Logic - Development Steps

Graphic Application Development

Development Steps

This development mode handles in standard the relationships between the various Logical Views used by your application.

	ENTITIES	TYPE	GENERATION COMMAND	DOCUMENTED IN [Title Nb. or Manual Ref.]
1. Description of constant data Storage Structure . **and, if Database:**	Data Element Data Structure Segment Database Block	depends on DBMS	**GSQ** if SQL database **GCB** if other type	Ref. DD SPE Ref. DD DSQ Ref. DD D_ _
2. Construction of Logical Views	Data Structure Logical View	**V** **V**	no generation	Chapter 2: Logical View
3. Specification of Elementary Business Components	Business Component	**S**	**GGS**	Chapter 3: Business Component
4. Business Component for Initialization/Termination of request on Folder (optional)	Business Component	**IT**	**GGS**	*Subchapter* Initialization/Termination Business Component in chapter 3
5. Error Server **Error Message File**	Business Component Business Components Dialogue	**E** **S**	**GGS** **GEC**	Chapter 4: Error Handling
6. Communications Monitor	Business Component	**FM**	**GGS**	Chapter 5: Communications Monitor (graphic applications)
7. Construction of Folder and generation of the Services Manager	Folder	**F**	**GGS**	*Subchapter* Folder in chapter 6
8. Description of Logical Views (optional) **Extraction >> Proxy**	Folder View 2 possibilities : extraction from Folder View or from Folder	**FV** **FV** **F**	**GVC** **GVC**	*Subchapte* Description of a Folder View in chapter 6

☞ Development steps involved in the production of the Client part are documented in the *Developer's Documentation / eBusiness and Pacbench/CS Applications: **Graphic Presentation***.

Particular Case: single-view development

This development mode corresponds to the development of an application that manipulates only one Logical View. It is referred to as 'single-view development' in the whole documentation.

	ENTITIES	TYPE	GENERATION COMMAND	DOCUMENTED [Title Nb. or Manual Ref.]
1. Description of constant data Storage Structure. **if Database:**	Data Element Data Structure Segment Database Block	depends on DBMS	**GSQ** if SQL database **GCB** if other type	Ref. DD SPE Ref. DD DSQ Ref. DD D_ _
2. Construction of Logical Views	Data Structure Logical View	**V** **V**	no generation	Chapter 2: Logical View
3. Specification of Business Components **. Extraction (>> Proxy)** **. Generation of Services Manager**	Business Component	**S**	**GGS** **GVC** **GGS**	Chapter 3: Business Component
4. Error Server **Error Message File**	Business Component Business Components Dialogue	**E** **S**	**GGS** **GEC**	Chapter 4: Error Handling
5. Communications Monitor	Business Component	**FM**	**GGS**	Chapter 5: Communications Monitor (graphic applications)

☞ Development steps involved in the production of the Client part are documented in the *Developer's Documentation / eBusiness and Pacbench/CS Applications:* ***Graphic Presentation***.

To implement an object-oriented application design, it is advised to define all processing associated with the same Logical View or at least its entry point, in the same Business Component.

Conversely, it is advised to define a Business Component for each Logical View, in order to facilitate the reuse and hence the maintenance of Logical Views.

TUI Application

	ENTITIES	TYPE	GENERATION COMMAND	DOCUMENTED IN [Title Nb. or Manual Ref.]
1. Description constant data Storage Structure **if Database:**	Data Element Data Structure Segment Database Block	depends on DBMS	**GSQ** if SQL database **GCB** if other type	Ref. DD SPE Ref. DD DSQ Ref. DD D_ _
2. Construction of Logical Views	Data Structure Logical View	**V**	no generation	Chapter 2: Logical View
3. Specification of Business Components	Business Component	**S**	**GGS**	Chapter 3: Business Component
4. Error Server **Error Message File**	Business Components Business Components Dialogue	**E** **S**	**GGS** **GEC**	Chapter 4: Error Handling
5. TUI Client	C/S Screen	**C**	**GGC**	Ref. DD OA

Development steps involved in the production of the TUI Client part are documented in the *Pacbench C/S - Business Logic & TUI Clients Manual (Ref. DD OA).*

It is advised to define a Business Component for each Logical View, in order to facilitate the reuse and hence the maintenance of Logical Views.

Although this type of development is not object-oriented, it is advised to define all processing associated with the same Logical View or at least its entry point, in the same Business Component.

Chapter 2: Logical View

Prerequisite: Description of Constant Data and its Storage Structure

External resources (constant data) are described in **Data Element**, **Data Structure** and **Segment** type entities.

You can import design specifications – entered in the Pacdesign WorkStation – to the Pacbench WorkStation and then complete them.

You will also describe **Database Blocks** if constant data is stored in a database (DBMS).

These descriptions are documented in the *Data Dictionary* and *Database Description* Manuals.

Data Elements Used in a Graphic Application

For a graphic application, the first character in the code of a Data Element called in a Logical View, User Buffer or Local Buffer must be a letter. Besides, **`A(n)`** or **`X(n)9(n)`** formatted and **`FILLER`**-type Data Elements are not authorized in a Logical View.

Definition of Data Structure and Logical View

Before defining a Logical View, you must determine which *Data Structure* it will be attached to.

If you have to define a new Data Structure, open the **`Definition`** window of a Data Structure occurrence (2-character **code**; the first being alphabetical, the second alphanumerical).

In this Definition, you enter:

- the **name** (30 characters maximum)
- the entity **type**

  `V` type of a Data Structure which groups Logical Views. This value is **required**.

  Once entered, this value cannot be modified.
- optionally, associated **keywords**.

Now open the **`Definition`** window of a Logical View occurrence (4-character **code**: **`DSXY`**, the first two characters correspond to the Data Structure code, if **`X`** is not **`0`**, **`Y`** can be alphanumeric, but if **`X`** is **`0`**, **`Y`** must be numeric).

If you press **`F1`** while the cursor is in an input field, on-line help is displayed; it contains a list of all possible values. If you double click on the desired value, it will be automatically entered in the field.

In this Definition, you enter:

- the **name** (36 characters maximum)
- optionally, associated **keywords**
- the **number of occurrences** (from `1` to `999`)

You must define the maximum number of occurrences that a Logical View can contain, i.e. its iterative capacity.

Actually, a Logical View groups all the business data of the message. Depending on the Client request and the answer of the Business Component, the message will have to include one or more occurrences.

Recommendation :

With an iterative capacity set to `1`, all queries processed by any Business Components associated with this Logical View are limited to one occurrence. In this case, these Business Components would be unable to perform the services of population selection and paging.

- the transfer direction

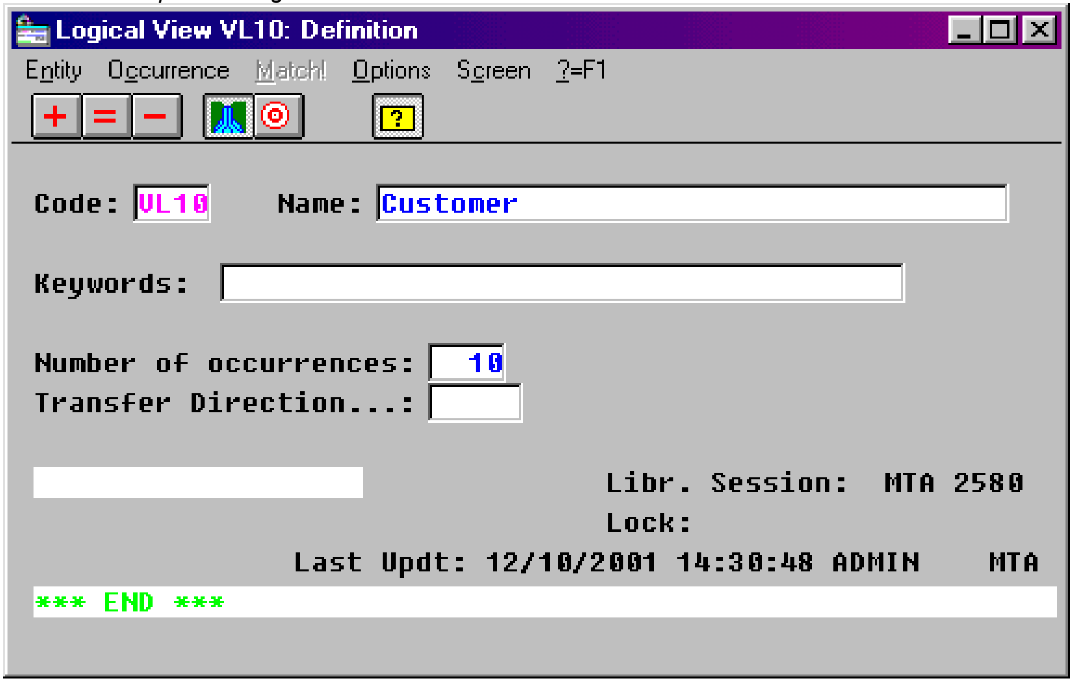
`blank` Default value
In both directions for an update service or a selection service.
It is the **required** value for graphic applications.

`C` Client to Business Component:
From Logical View to Segment for an update service.

`S` Business Component to Client:
From Segment to Logical View for a selection service.

☞ You enter the `C` or `S` value in this field only if the transfer direction is the same for all the Data Elements. Otherwise, you must specify this information for each Data Element in the Logical View's `Call of Elements` window.

Example of a Logical View Definition:



Code: VL10 Name: Customer

Keywords:

Number of occurrences: 10
Transfer Direction...:

Libr. Session: MTA 2580
Lock:
Last Updt: 12/10/2001 14:30:48 ADMIN MTA
*** END ***

Description of the Logical View

Logical Views are described in the `Call of Elements` window.

If you press `F1` while the cursor is in an input field, on-line help is displayed; it contains a list of all possible values. If you double click on the desired value, it will be automatically entered in the field.

List of Fields or Call of Elements

To describe the contents of the Logical View occurrence is to list the fields making up this occurrence. Each field corresponds to a Data Element defined in the VisualAge Pacbase Dictionary.

Open the `Call of Elements` window of the Logical View occurrence and, for each called Data Element, enter:

- a **line number** (`LIN` column), which identifies the call of the Data Element
- the **code of the Data Element** (`ELEM` column)
- the Data Element check option (`P` column):

`O` Required Data Element. The check for its presence is generated in the Client, and in the Business Component if the option `CHECKSER=YES`.

☞ Checks specified in the Data Element's characteristics are also generated.

`p` Required Data Element. The check for its presence is generated only in the Business Component, if `CHECKSER=YES`.

☞ Checks specified in the Data Element's characteristics are also generated.

`blank` Optional Data Element. Checks specified in the Data Element's characteristics only are generated in the Client, and in the Business Component if `CHECKSER=YES`.

`F` Optional Data Element. Checks specified in the Data Element's characteristics are generated in the Business Component only, if `CHECKSER=YES`.

Graphic Applications - Specifics

- A logical View can only call Data Elements whose code starts with a letter or a number.
- When the Data Element code starts with a number, this latter is automatically prefixed by `dataelement` at generation.
- Data Elements called in a Logical View cannot be repeated.
- Group Data Elements of a Logical View are not taken into account during Proxy generation.
- `A(n)` or `X(n)9(n)` formatted and `FILLER`-type Data Elements are not authorized in a Logical View.
- Data Element redefinition is not authorized in a Logical View.

Identifiers

Among the data which describes an information occurrence, you must determine which are the **identifiers**, i.e. Data Elements which will be used as selection criteria or start key for the reading of data occurrences in the database.

In the list of Data Elements of the Logical View, you specify identifiers by assigning them the **`U`** type (**`TYPE`** column).

In the COBOL generated code, the description of the Logical View identifiers is as follows:
`1-VIEW-DELCO`

Example

`1-LV01-NOCL` for the `LV01` Logical View where the `NOCL` Data Element has a `U` type.

Identifiers allows automatic paging functions (list continuation).

Graphic Applications - Specifics

- An identifying Data Element must never be blank.
- For a depending Logical View (see Description of a Folder), identifier(s) common with those of *all parent Logical Views* must be entered in the same order.

Extraction Method Parameters

When a Client calls a selection service, this service may implement an **extraction method** and its **parameters by value**.

The Business Component which answers the call must recognize the extraction method and its parameters to correctly perform the service. The only way to send this information is via the message sent by the Client.

- The extraction method is sent to the Business Component via the technical part of the message.
- Parameters must be described in the application part of the message, i.e. in the Logical View.

In the **`Call of Elements`** window, you specify the parameters by value for *all* the extraction methods which may be used by the Logical View, by calling Data Elements defined in the VisualAge Pacbase Dictionary. For each parameter, you enter:

- a **line number** (**`LIN`** column), which identifies the call of the Data Element,
- the **code of the Data Element** (**`ELEM`** column).
- the **`E`** **type** (**`TYPE`** column), which indicates that it is a parameter.

Each identifier-Data Element (**`U`** type) is an implicit parameter. The other Data Elements that describe the contents of the Logical View cannot be used as parameters. The **`E`**-type Data Element does not belong to the description of a Logical View occurrence; if necessary, you can define and call a child Data Element.

In the Business Component's generated COBOL code, the name of the corresponding **`WORKING`** area is as follows:
`1-VIEW-DELCO`

Example

***1-LV01-LNCLA** for the **LV01** Logical View where the **LNCLA** Data Element has the **E-** type.*

👓 For more information on extraction methods, refer to section Extraction Methods.

Data Transfer

- To implement a selection service, data must be transferred from the Segments to the Logical View, once accesses to external resources have been specified.

  👓 The specification of a selection service is documented in *Subchapter* Selection Service.

- When implementing a *check and update* service, data must be transferred from the Logical View to the Segments, after specifying accesses to external resources for checking and before specifying the access for updating.

  👓 The specification of a check and update service is documented in *Subchapter* Check / Update Services.

These data transfers are automatically generated by mapping the Data Elements of the Logical View with the Data Elements of the Segments.

You specify this mapping in the **Call of Elements** window of the Logical View.

For each field in the Logical View, i.e. for each Data Element, you enter:

- the mapped Segment field (**TRANSFER** column), made up of the Segment code and the Data Element code if the latter is different from the Logical View's Data Element code.

  Both fields mapped must be in the same format. One same Data Element can feed several indexed or table Segments.

  ☞ **TUI client Specifics**: Logical View data cannot be transferred to a Segment which belongs to both *Top* and *Bottom screen* categories.

- the operator (**OPER** column) corresponding to the transfer mode (COBOL verb which will be generated):

blank or **M**	substitution (move)
+	addition (add)
-	subtraction (subtract)

- the data transfer direction (**DIR** column):

blank Default value
In both directions for an update service or a selection service.

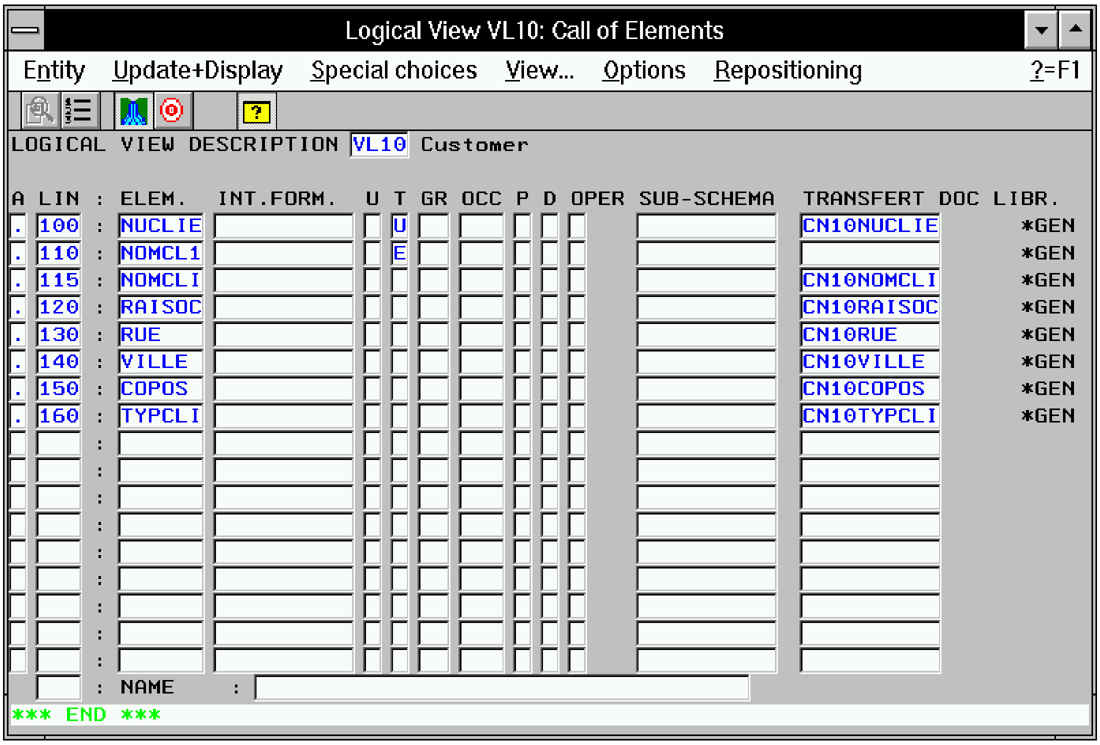
☞ This value is **required** for a graphic application.

C Client to Business Component:
From Logical View to Segment for an update service.

S Business Component to Client:
From Segment to Logical View for a selection service.

☞ You enter the **C** or **S** value in this Data Element only if the corresponding field in the Logical View's **Definition** window is blank.

Example of a Logical View Call of Elements:



```
LOGICAL VIEW DESCRIPTION VL10 Customer

A LIN : ELEM.  INT.FORM.  U T GR OCC P D OPER SUB-SCHEMA  TRANSFERT DOC LIBR.
. 100 : NUCLIE              U                             CN10NUCLIE     *GEN
. 110 : NOMCL1                E                                          *GEN
. 115 : NOMCLI                                            CN10NOMCLI     *GEN
. 120 : RAISOC                                            CN10RAISOC     *GEN
. 130 : RUE                                               CN10RUE        *GEN
. 140 : VILLE                                             CN10VILLE      *GEN
. 150 : COPOS                                             CN10COPOS      *GEN
. 160 : TYPCLI                                            CN10TYPCLI     *GEN
      :
      :
      :
      :
      :
      :
      :
      :
      :
      : NAME    :
*** END ***
```

Chapter 3: Business Component

Definition and Options

Dialogue Level

The Business Component Dialogue is a logical envelop which groups the various Business Components of an application.

In the Business Component Dialogue, you give the general characteristics of the application and specify generation variants and options which will apply by default to all attached Business Components.

In the **Dialogue entity**, you create and describe the Business Component Dialogue of the application.

Definition

The `Definition` of a Business Component Dialogue consists in creating an occurrence of the Dialogue entity (2-character **code**).

Data entered in the Dialogue is used as default data at the Business Component level.

The required fields for the Dialogue are the following:

- the **name**
- the **type** (optional at the Dialogue level; however, it is recommended to use the `S` -type for Business Component).

☞ The deletion of a Dialogue is possible only if there is neither a Business Component nor a Dialogue description attached to it.

A `Name of the PSB or Sub-Schema` field in the `Dialogue Complement` window should also be entered but in some cases only (DL/1 Database for example). For the IMS variant, this field contains the application PSB code.

Setting Generation Options

A number of options are needed when generating Business Components.

All Business Components of a given application must have the same generation options. It is then *recommended* to centralize these options in the Dialogue; they are automatically used when each attached Business Component is generated. Otherwise, make sure that each Business Component has the same options.

You enter these options in the Dialogue `Generation Parameters` window, on `O`-type lines, as follows:

`OPTION=option value` (no space before and after `=`).

Several options, separated by a blank, can be entered on each line.

No check is performed when you enter generation options. But, if these options are not standard options, messages are displayed in the generation report.

⚠ Other options being specific to each Business Component, it is then necessary to enter them at the level of each component. For more details, see section.

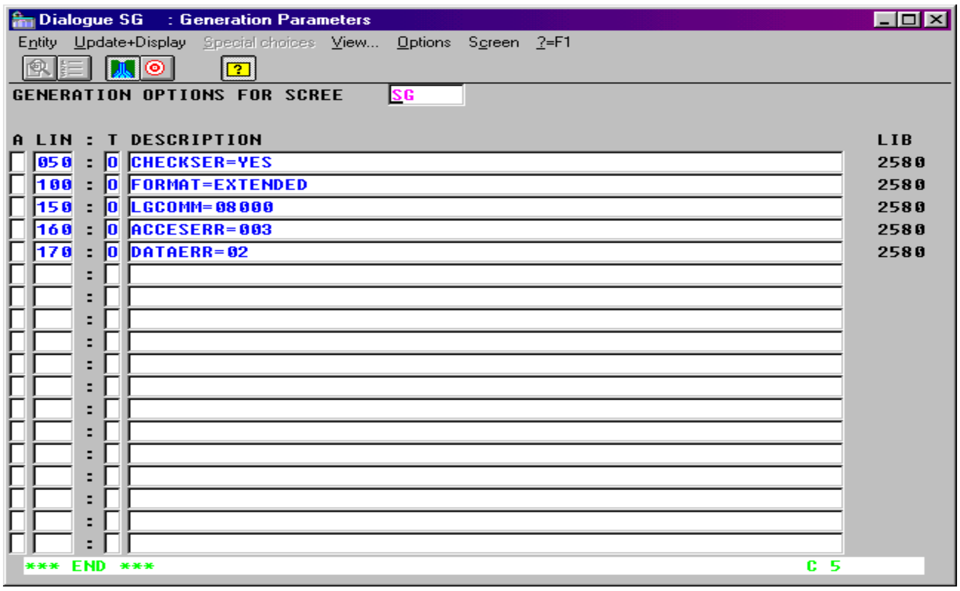
General Options

ACCESERR	Maximum number of errors on Segment access that can be returned to the Client. 3-character value Default value = 001		
DATAERR	Maximum number of errors on the Logical View's Data Elements that can be returned to the Client, 2-character value Default value = 01		
CALLTYPE	**Target**	**GUI**	**TUI**
	CICS	LINK* CALL	LINK* CALL
	DPS7	CALL*	CALL*
	DPS8	LINK* CALL	LINK* CALL
	IMS	CALL*	CALL* CHNG
	MICROFOCUS	CALL*	CALL*
	TANDEM PATHWAY	PATHSEND	
	TUXEDO NB : default value = TPCALL value **not** to be entered	CALL	CALL
	* : default value		
CHANGE	Only with TUXEDO: Default value: NO Call of the Business Component with the TPNOCHANGE parameter of the CALL If CHANGE=YES: call of the Business Component with the TPCHANGE parameter of the CALL It is strongly recommended to code the same option in the Folder Dialogue Generation Parameters screen, if implemented.		
CHECKSER (1)	Default value: CHECKSER=NO: the Business Component does not check the Logical View data CHECKSER=YES: the Business Component checks the Logical View data		
CONNECT	Management of the connections to and the disconnections from SQL databases Default value: CONNECT=YES: (dis)connections generated in the Communications Monitor For TUXEDO, these (dis)connections are generated in the Initialization and Termination Business Components (**SI** or **ST** type). In this case, you must enter in their Call of Segments (semi local) window, one of the Table Segments from the database you want to be (dis)connected to. CONNECT=NO: Value only used for TUI applications (dis)connections automatically generated in the Business Component		
FORMAT	For graphic applications, you must code the value EXTENDED		
LGCOMM	Length of the communication area between: . the Communications Monitor (graphic application) and the Business Component, or . the Folder and the Business Component or . the TUI Client or Client Monitor and the Business Component. 5-character value, **required if several Business Components are attached to the Dialogue.**		
NULLMNGT (1)	Default value: NULLMNGT=NO: No management of the presence of the Logical Views' Data Elements at the Proxy level NULLMNGT=YES: Management of the presence of the Logical Views' Data Elements at the Proxy level NB: If NULLMNGT=YES, the VECTPRES=YES option is systematically implemented.		

NUVERS	Version number of the Business Component, the Proxy, and TUI Client. This option allows to transfer to the Business Component, via the technological buffer, the piece of information that will handle the various versions of Client and Business Components. Default value for TUI client only: VisualAge Pacbase generation session
PACTABLE	VisualAge Pacbase code of the Pactables access Program to the CICS Business Component. Default value: **PACTABLE**
SERVBUFF	Available only with specification of a Folder This option identifies the data structure which defines the Server Buffer. **SERVBUFF=xx** **xx** corresponds to the data structure code (2-character code). At the level of *each Business Component,* various segments of this data structure may be selected via the **SEGMBUFF= s1s2s3s4**... option. For more details on the Server Buffer, refer to section Server Buffer.
TRAN	Only with TUXEDO Default value: **TRAN=YES**: Use of the transactional mode for the TUXEDO variant. **TRAN=NO**: No use of the transactional mode It is strongly recommended to code the same option in the Folder Dialogue, if implemented.
VECTPRES (1)	Default value: **VECTPRES=NO**: No management of the presence of the Logical Views' Data Elements **VECTPRES=YES**: Management of the presence of the Logical Views' Data Elements NB: If **CHECKSER=YES**, the **VECTPRES=YES** option is systematically implemented.
CURSUFF	Default value: **CURSUFF=NO**: the name of the cursors are not suffixed **CURSUFF=YES** the name of the SQL cursors are suffixed with the corresponding VisualAge Pacbase screen code.

(1) Graphic application: **VECTPRES** and **CHECKSER** options in the Business Components which call the same Logical Views must be identical.

Example of the ***Generation Parameters*** *window of a Business Component Dialogue:*


Dialogue SG : Generation Parameters
Entity Update+Display Special choices View... Options Screen ?=F1

```
GENERATION OPTIONS FOR SCREE    SG

A LIN : T DESCRIPTION                                          LIB
  050 : 0 CHECKSER=YES                                         2580
  100 : 0 FORMAT=EXTENDED                                      2580
  150 : 0 LGCOMM=08000                                         2580
  160 : 0 ACCESERR=003                                         2580
  170 : 0 DATAERR=02                                           2580

*** END ***                                             C 5
```

Additional Options for Single-View Development

ERRLAB	This option specifies how the errors must be returned. By default, ERRLAB=YES : the Communications Monitor retrieves the key, gravity and error messages returned by the Business Component before sending back the answer to the Client component. But if the option value is set to NO, the error key only will be returned. In that case, the error message is locally read by the client application.
ERRSERV	VisualAge Pacbase code of the Error Server. **This option is required**.
LOCATION	**This option is required**. This option represents the VisualAge Pacbase code of the Communications Monitor (on 6 characters) followed by the environment name (on 20 characters maximum). This option must be left adjusted. Example: LOCATION=FMCODE-NOMENVIR If you use more than one LOCATION, enter each of them on a separate line, all left adjusted. The positioning of this option triggers the generation of two COBOL sources: the first one corresponds to the generation of the Business Component selected and the second one corresponds to the Services Manager source (it is a purely technical component which is used to generate all the facilities usually generated by the Folder). You can inhibit one of the two generations from the Business Component Definition. (For more details, refer to section Definition of the Business Component).
LOCKMODE	Option used to specify the type of lock. The lock process prevents the update of a set of data in order to avoid inconsistency. (This service is documented in section Lock / Unlock Service). If the value is NO (default) no lock is specified. The PESSIMIST value creates an exclusive appropriation of the occurrence which is locked to all other users during updating. The OPTIMIST value performs the lock only when the update is requested and compares the initially loaded Folder image with its current image in the database. If they are identical, the update is accepted.
PAGEMODE	Option specifying the paging mode. The NOEXTEND value (by default) allows to page forwards and backwards through a predefined population. Each paging operation (forwards or backwards) executes a read request and its result replaces that of the previous read. The EXTEND value allows you to page forwards through a predefined population, and the result of each paging adds to the preceding read. The reading of the previous pages is then handled by the graphic client.

Additional Options for TUI Mode Development

MONSER	VisualAge Pacbase Code of the Server Monitor.

User Buffer

The message between Client and Server components can carry contextual data related to the execution of the called service.

The definition of this data structure and the processing of its content are the Developer’s responsibility. This data must be defined in a structure called a *User Buffer*.

It allows you to position additional fields as complements to the technical Buffer (data compression, identification of the end user, etc).

This data structure is included in each message exchanged between all the Dialogue Business Components and the Client components.

If your application is to use a User Buffer, you must declare the corresponding Segment occurrence in the Business Component Dialogue, with a **`W`** organization.

This Segment and its organization must be entered in the **`Call of Segments (semi local)`** window, in the Dialogue box opened via the choice **`Unused Segments`** of the **`View`** menu.

Business Components of a Folder (*root and depending nodes*) must use the same Segment to define a User Buffer[1].

The User Buffer of a Business Component called by or associated with a *reference node* must have the same length as that of the Buffer used for the root or depending nodes.

Graphic applications :

- A User Buffer contains neither repeated nor **`FILLER`**-type Data Elements.

Server Buffer

It is sometimes necessary and sufficient to share data exclusively between the Business Components, whether they are called by the same Services Manager or called between themselves by the mechanism for call of second-level Business Component.

With the User Buffer, you can define a memory area shared between the Business Components which participate in the execution of a request.

The declaration of the Server Buffer is made at the level of the **`Generation Parameters`** of the Dialogue window (or Business Component) with the option :

`SERVBUFF=xx`
where **`xx`** identifies the Data Structure.

At the level of each Business Component, various segments of this Data Structure may be selected via the option :

`SEGMBUFF=s1s2s3s4`… where **`s1`**, **`s2`**, **`s3`** … represent the last two characters of the Segment code.

For more details on the **`SEGMBUFF`** option, refer to section Business Component Options. If this option is not coded, all the Segments of the Data Structure are selected.

- **Use rules :**
  - All the Business Components of a Folder – *whatever the node type* – must use the same Data Structure to define a Server Buffer.
  - This rule applies also to a *Business Component called* by another using a Server Buffer. It must also use the same Server Buffer.

[1] This rule will be automatically respected if these Business Components belong to one same Dialogue (recommendation made at the level of a Folder construction).

  - A Services Manager does not need any particular option to call a Business Component using a Server Buffer. However, each modification of the Server Buffer structure implying a modification of its maximum length requires the regeneration of the Folder (Services Manager) attached to the Business Components concerned.
  - A TUI Client calling a Business Component which uses a Server Buffer must go through the intermediary of the Server Monitor.

- **Technical information on the generation of the Server Buffer** :
  - Data Structure Segments are generated as a redefinition of one another. If a '`00`' Segment exists, it is considered as a common part of all the others.
  - The group fields, `occurs`, `redefines` are taken into account.
  - The `FORMAT=EXTENDED` option which triggers the generation of the fields in extended `DISPLAY` use is ignored. Binary and packed fields are thus generated with the corresponding uses and the sign in the numeric fields is virtual.
  - An alignment `FILLER` is generated for all the selected Segments which have a length smaller to that of the longest Data Structure Segment (even if the Segment is not selected, this in order to have a constant length for all the Business Components whatever the Segments selected on each of them).

Business Component Level

Definition of the Business Component

To define a Business Component is to create an occurrence of the Business Component entity (6-character **code**, the first two being identical to those of the Dialogue code).

In the `Definition` window, you enter:

- the **name**
- optionally, one or more **keywords**
- the `S` **type**
- the generation **variants**
- the **transaction** code (TUI only; needed when the Client component calls the Business Component via a transaction code)
- the **commands in front/in back** (`CCF`/`CCB`)

In a single-view development, you can inhibit the generation of the:

  - Business Component, you just need to enter the value `$` in the commands `CCF (Commands in front)`, on the `Program` line.
  - Services Manager, you just need to enter the value `$` in the commands `CCF (Commands in front)` on the `Complement` line.

- the external name.
  - the `Program` field is used to specify the external name of the generated Business Component (default=Business Component code).
  - For a single-view development, the `Complement` field is used to specify the external name of the Services Manager (default value = Business Component code).

You must enter a value which is different from the Business Component code in one of these fields at least.

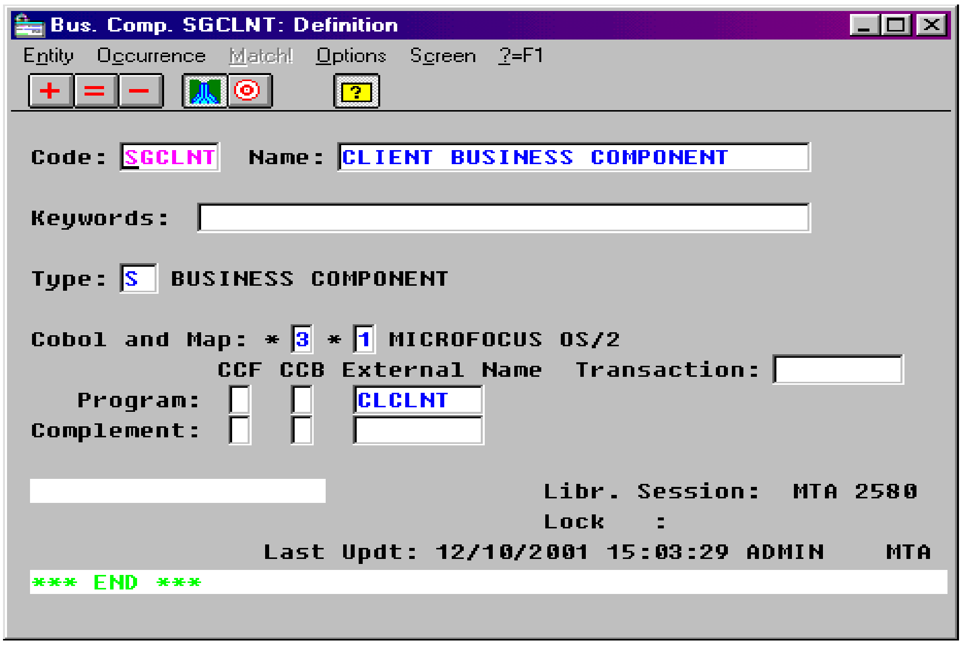
☞ If you use Tuxedo with a version earlier than 6.2, you must enter the code of the View in this field. The single-view development is hence prohibited for Tuxedo in a version earlier than 6.2.

Therefore, the same Business Component can be used for all modes of development. You have just to make sure that the options required for the selected mode are correctly specified.

☞ The deletion of a Business Component is possible only if there is no description of the Business Component.

💡 If you press **F1** while the cursor is in an input field, on-line help is displayed; it contains a list of all possible values. If you double click on the desired value, it will be automatically entered in the field.

Example of a Business Component Definition



```
Code: SGCLNT   Name: CLIENT BUSINESS COMPONENT

Keywords:

Type: S  BUSINESS COMPONENT

Cobol and Map: * 3 * 1 MICROFOCUS OS/2
                     CCF CCB External Name  Transaction:
   Program:                  CLCLNT
Complement:

                                 Libr. Session:  MTA 2580
                                 Lock    :
                      Last Updt: 12/10/2001 15:03:29 ADMIN    MTA
*** END ***
```

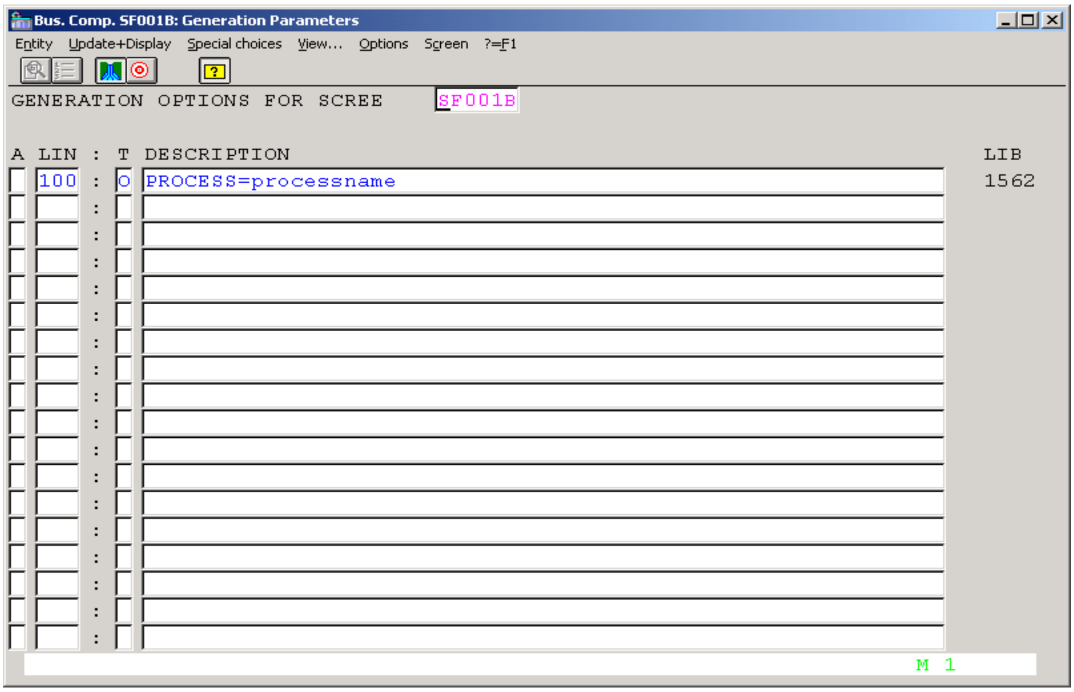
Business Component Options

Generation variants and options are by default those of the Business Component *Dialogue* and are then documented in section Dialogue Level.

However, various options are necessarily specified at the Business Component level in the **Generation Parameters** window (**O-** type line).

PROCESS	Available for TANDEM PATHWAY only. Name of the Process when calling the Business Component. 15 characters maximum without space. No lowercase/uppercase conversion.
SEGMBUFF= s1s2s3s4...	Available only with specification of a Folder. Option used for the Server Buffer. It allows to select Segments of the Data Structure that defines the Server Buffer. **s1**, **s2**, **s3**....represent the first two characters of the Segment code. 10 Segments may be positioned by the **SEGMBUFF** option. A maximum of 10 **SEGMBUFF** options may be called at the level of each Business Component. The '**00**' Segment does not need to be selected, it is systematically generated if it is present in the Data Structure. Default value : All the Data Structure Segments are selected. For more details on the Server Buffer , see section Server Buffer.

Example of a Business Component ***Generation parameters*** *:*


Bus. Comp. SF001B: Generation Parameters
Entity Update+Display Special choices View... Options Screen ?=F1
M 1

```
GENERATION OPTIONS FOR SCREE   SF001B

A LIN : T DESCRIPTION                                   LIB
  100 : O PROCESS=processname                           1562
```

Global Structure of the Business Component

Using the information you have entered in the Business Component, the generator produces a set of functions by which service calls on Logical Views will be answered.

```
05        BEGIN SERVER

05        SQL DECLARATIONS

05        LOGICAL VIEW PROCESSING
   10               Begin Logical View
   10               Check / Update services
   10               Selection service
   10               User service
   10               Lock service
   10               Unlock service
   10               Error handling
   10               End Logical View

05        END SERVER

   10               Data transfer: Logical View to Segment
   10               Data transfer: Segment to Logical View
   10               Logical accesses to Segments for check
   10               Logical accesses to Segments for update
   10               Logical accesses to Segments for selection

05        PHYSICAL ACCESSES TO SEGMENTS

05        PERFORMED GENERAL PROCESSING
```

This structure gives a global view of the generated program.

☞ A summary of the Business Component generated in standard is available in the appendix at the end of the manual.

Begin Server

Retrieves the information contained in the communication area and carries out initializations.

SQL Declarations

Describes the clauses required for the SQL declarations (**WHENEVER** and **DECLARE CURSOR**).

Logical View Processing

If several Logical Views are declared in the Business Component, there will be as many blocks as Logical Views.

The processes are sequentially performed. They include the possible services for the Logical View, i.e.:

- for services generated automatically :
  - check and update
  - selection
- other services :
  - User Service
  - call of a second-level Business Component
  - (un)lock

The following table lists the generated services from the type of use specified.

☞ The different types of use are documented in details for each type of service in the following subchapters:

- Check / Update Services
- Selection Service
- User Service.
- Call of a Business Component by another.
- Lock / Unlock Service.

The services are presented in the order in which they appear in the generated code.

Use in reception :	**Generated service :**
E, C, M, S, T, X	E (check)
L, C, M, S, T, X	L (read)
C, M, S, T, X	M (check + update)
C, M, S, T, X	T (check + update + selection)
E, C, M, S, T, X	X (check + selection)
Use in display :	**Generated service :**
A, T, X	A (selection)

For example, if a *C* use type is specified on a Segment, the following services are automatically generated :

- check,
- read,
- check + update,
- check + update + selection,
- check + selection.

However, only one of the generated services will actually be executed, according to the Business Component request on the Logical View.

The Client component can only request the execution of one of the services specified in the Business Component.

For example, a Client component cannot request a check/update service on a Segment whose type of use in the Business Component is **L**.

End Server

Returns the information to the Services Manager (or the Client or Client Monitor for a TUI development).

Elementary Procedures

The **elementary procedures** are executed by individual `PERFORM` from the 'Logical View' level. Two types of procedures are performed:

- Data transfer

  The data transfer direction (Logical View to Segment or Segment to Logical View) depends on the value specified in the `Call of Data Elements` of the Logical View, in the field `DIR`.

  Depending on the services, one or the other direction or both directions will be actually executed.

  For example, if the `DIR` field contains no value (default), both directions will be generated. But if the requested service is a check and update service, only the transfer direction from Logical View to Segment will be executed.

- Logical access to Segments, to perform services requested by the Client component.

The following table illustrates the elementary procedures (data transfer direction and logical accesses) executed for each service.

	Elementary procedures				
Generated service	**T1**	**T2**	**T3**	**T4**	**T5**
`E` (check)			*		
`L` (read)		*			*
`M` (check + update)	*		*	*	
`T` (check + update + selection)	*	*	*	*	#
`X` (check + selection)		+	*		+
`A` (selection)		*			*

P1 Data transfer from Logical View to file or table Segments

P2 Data transfer from file or table Segments to Logical View

P3 Check sequence

P4 Update sequence

P5 Selection sequence

+ The **X** service (check + selection) is generated according to the use in display. If the use is **A** (read with selection of information), the **P5** processing is called with a management of the check return code), followed by the **P2** processing. Otherwise, only the **P2** processing is performed.

For the T service (check + update + selection), the P5 processing is called either in the processing in selection of the Logical View, or directly called after the update sequence ('update done'). The 'update done' is generated when the Segment is not used in a processing for display. It means that the read will only concern the modified line.

Physical Accesses

The **physical accesses to data** make up a third processing level. A physical access is performed for each Segment logical access. The generated physical access depends on the constant data storage structure.

Changing the physical storage mode of data does not modify the logical structure of the Business Component.

Inserting Specific Code

General Principles

The Business Component is structured in functions, generated either through standard processing or from called macro-structures and/or specific code, such as a conditioned access to data or data transfers.

Before inserting specific code, you **must be familiar with the structure** of the Business Component and respect it, which will also facilitate its maintenance.

👓 This structure is documented in *Subchapter* Global Structure of the Business Component.

Specific code is written in the **`Local Specific Code`** window of the Business Component.

👓 The use of this window and its dialog boxes is documented in a specific chapter of the *VisualAge Pacbase WorkStation* Reference Manual. Complete documentation on input fields and their possible values is found in the *Structured Code* and *Pacbench C/S - Business Logic & TUI Clients* Reference Manuals.

A Business Component's specific procedure is always inserted **relatively to** standard processing:

- before a standard procedure,
- at the beginning of a standard procedure,

- at the end of a standard procedure,
- after a standard procedure,
- in replacement of a standard procedure.

This allows you to use the standard procedures generated by VisualAge Pacbase while adapting them to your needs.

The Structured Code notions of functions, sub-functions, hierarchical level, structure type, and conditioning also apply here.

As a reminder, you must know that functions always have a level `05`. Sub-functions have a level `10` by default but they can have levels `06` to `98`.

Important principle:

A sub-function is lower than another one if its level number is higher.

Example:

In a function, a sub-function with a level `15` is dependent on the sub-function with a level `06` to `14` which precedes it.

A sub-function included in another one is executed only if the higher function is executed.

Depending on the nature of the specific procedure, you will use one of the following three types of insertion:

- relative to standard procedures with an `05` hierarchical level, i.e. the blocks corresponding to the beginning and the end of Business Component, and the Logical View blocks,
- relative to the standard procedures of a Logical View, whether in the program body or in the performed elementary processing,
- relative to the physical access procedures to Segments, i.e. in function `80`.

Important Rules

- You cannot create two sub-functions at the same insertion point (same generated function) with the same type of block.
- Each insertion/replacement of procedures on a Logical View (check and update, selection, transfer, etc.) is limited to 36 sub-functions.
- The replacement of a check on Data Element or the insertion of such a check is limited to only one sub-function per Data Element and 90 lines of generated code for each of these sub-functions.

☞ These checks are generated if the option **`CHECKSER=YES`** is present in the **`Generation Parameters`** of the Dialogue (or Business Component).

No check is generated on repeated elementary Data Elements.

This option also conditions the transfer of data from the Logical View to the SQL Segments depending on the Data Element presence.

- The replacement of a procedure deletes the automatically generated tag. To make this tag appear again, code the **Y** operator on the next line.

- In a Business Component used by a graphic application, no provoked ABENDs are authorized.

Insertion Relative to the 05 Hierarchical Level Processing

In the program body, several blocks with a 05 hierarchical level are generated:

- 1 block for the Server initialization **FSERVER**
- n blocks for the Logical View **FVIEW**
- 1 block for the Server end **FSERVER-END**

To add a specific procedure, insert your procedure relative to one of these blocks: before, after, or in replacement of.

To do so, create a function by entering:

- a 2-character **code**, free (except 80 and 81),
- a **title** (**N** operator in the **OPE** column),
- an 05 **hierarchical level**: only possible level for a function,
- a ***C structure type** (**TY** column),
- an **insertion point** (**CONDITION** column), which includes:
  - the codes (up to 4) used to identify the position of your insertion relative to the standard processing.

These codes vary, depending on the services. They are then documented in the paragraph *Inserting Specific Code* of each service.

  - the type of setting:
    A before (ante),
    P after (post),
    R in replacement of.

Example:

Insertion after the block 'Begin Server' to carry out the initializations before the Logical View processing:

```
OPE OPERAND                               TY CONDITION
Function IN                               Level 05
 N   INITIALIZATIONS                           *C BS P
```

Insertion Relative to the Logical View Processing

In the Logical View block, either in the program body or in the performed elementary levels, you find all the standard procedures related to the implementation of this Logical View's services. Each one of these standard procedures is associated with only one Logical View.

This section presents the principles specifically applied to each service.

- To add specific procedures to a Logical View, you must first identify it. To do so, create a function by entering:

    - a 2-character **code**, free (except `80` and `81`),
    - a **title** (`N` operator in the `OPE` column),
    - an `05` **hierarchical level** identical to that of the Logical View block,
    - a `*C` **structure type** (`TY` column),
    - an **insertion point** (`CONDITION` column) which specifies the code of the Logical View affected by the specific procedure (4 characters).

This function must only contain the line `N`. No procedure or comment can be inserted at this level.

- Then, you position each specific procedure to be added, relative to a standard procedure of the Logical View. Create a sub-function, defined by:
    - a 4-character **code**, the first two identifying the function, the following two being free,
    - a **title** (`N` operator in the `OPE` column),
    - a **hierarchical level** equal to or greater than that of the standard procedure, necessarily higher than `05` and lower than `99`,
    - the **type of setting** (`TY` column) by using one of the following structure types:

      `*A` before (ante) the tag of the standard procedure.

      `*B` beginning (begin) of the standard procedure, just after the beginning tag.
      This setting is possible for performed procedures only.

      `*P` after (post) the standard procedure:

      * just before the end tag, if the hierarchical level is higher than that of the standard procedure,

      * just after the end tag, if the hierarchical level is equal to that of the standard procedure.

      `*R` in replacement of.
    - an **insertion point** (`CONDITION` column) where you indicate the codes (up to 4) used to identify the position of your insertion relative to the standard processing.

These codes vary, depending on the services. They are then documented in the paragraph *Inserting Specific Code* of each service.

- Then you write the specific code, according to the rules of the **Structured Code**.

Insertion Relative to the ‘Physical Accesses’ Level

Physical accesses are generated in function `80`.

The function `80` of the Business Component is structured in sub-functions: one sub-function per Segment and per access type.

Specific procedure in function `80` is inserted by relative positioning:

- with an `*R` structure type, to replace a standard sub-function,
- with a `*C` structure type, to complete or create an SQL access.

This type of insertion is documented in section Customized SQL Accesses.

Replacing a Standard Sub-function

To replace a standard sub-function, you create a specific sub-function defined by:

- a 4-character **code**, the first two being 80 and the last two being non-numerical,
- a **title** (N operator in the OPE column),
- a 10 **hierarchical level**: identical to that of the standardly generated sub-function,
- an *R **structure type** to replace a standard sub-function by a specific sub-function (TY column),
- an **insertion point** (CONDITION column) which indicates the code of the accessed Segment possibly followed by the access type:
  - if the Segment code is followed by the access type, only the sub-function corresponding to the access type is replaced,
  - if the access type is not specified, all the access sub-functions to this Segments will be replaced.

Customized SQL Accesses

Principles

With relational databases, you can customize SQL accesses in order to:

- add or replace clauses in a standardly generated access or create a new physical access.
- take into account the selection criteria used by an extraction method.

The specification of these types of accesses (in the Segment's Generation Lines window) is detailed in the *SQL Database Description* Manual.

In any case, to implement these types of accesses in the Business Component, you must create a sub-function in function 80 by entering:

- a 4-character **code**, the first two being 80,
- a **title**, required (N operator in the OPE column),
- a 10 **hierarchical level**: identical to that of the standardly generated sub-function,
- a *C structure type (TY column),
- an **insertion point** (CONDITION column) which indicates:
  - the code of the relational table Segment,
  - the standardly generated access type, if you want to complete it with a customized access,
  - the code of the customized access to be taken into account,
  - the code of the extraction method (necessarily entered in *uppercase*), if selection criteria are to be taken into account.

Extraction Methods

- Description (Reminder)

All extraction methods likely to be used in a selection service call must be described (name and SQL implementation) in the Generation Lines window of the relational database-type Segment.

Four access commands are then necessary:

- `declare cursor`
- `open`
- `fetch`
- `close`

To generate these commands, you partially describe only one **EX** customized access, with the following clauses:

```
G   SQL         EX    EX    METHODNAME
G   WHERE ...
G   ORDER       ...
```

The extraction is carried out according to one or more criteria indicated in the **WHERE** parameter (and according to the sort indicated in the **ORDER** parameter).

You can customize the **DECLARE CURSOR** order by using other parameters. For more information, refer to the *SQL Database Description* Manual.

The selection parameter(s) must also be specified in the Logical View since they belong to the message sent to the Business Component.

For more information, refer to section Extraction Methods.

- **Use**

You must explicitly declare this customized access in function `80`.

The coding of this access request is presented in section Principles (Customized SQL Accesses).

The four physical accesses will be generated with the following tags:

F80-SEGT-DC-METHODNAME	for **DECLARE CURSOR**
F80-SEGT-P- METHODNAME	for **OPEN**
F80-SEGT-RN-METHODNAME	for **FETCH**
F80-SEGT-CL-METHODNAME	for **CLOSE**

In the graphic Client, the code of the extraction method is automatically loaded.

However, in a TUI client, you must specify the code of the extraction method in specific code. This code will then be transmitted to the Business Component via the technological buffer.

- **PERFORM** of the customized access

In the standard procedures, the **PERFORM** of a physical access is generated for each logical access to a Segment.

Logical access to a Segment is broken down as follows:

key loading	tag **FSEGT-SLCT-CATR-ALIM**
PERFORM of the physical access	tag **FSEGT-SLCT-CATR-CALL**
error handling	tag **FSEGT-SLCT-CATR-ERRS**

The **PERFORM** of the physical access is executed only when no extraction method is called.

If an extraction method is called, the **PERFORM** of the standard physical access is not executed, you must write the **PERFORMs** of the corresponding physical accesses by inserting specific code just after the standard **PERFORM** processing, conditioned by the method code.

Check / Update Services

Check / update services are services in reception: they retrieve the data entered in the Client component (via the Logical View) and compares it to the data in the base (via a Segment) for check or update.

- An **update** creates, modifies or deletes a Logical View instance in the database.

- There are **several types of checks**:
  - ▪ the checks that are specific to Data Elements and automatically generated from their characteristics specified in the windows :

    Definition : a date Data Element format, value type (numerical), …

    Description (Values …) : or range of values
  - ▪ the checks for the presence of Data Elements,

  These first two types of checks are standardly generated by positioning Business Components in the **Generation Parameters** screen of the Dialogue :

  - ➢ option **CHECKSER=YES**, for the checks specified at the Data Elements level,
  - ➢ option **VECTPRES=YES** to handle presence indicator for Data Element. (If the **CHECKSER** option is set to **YES**, the **VECTPRES** option is automatically **YES**)

  For each Data Element of the Logical View, you ask or do not ask for these checks in the **Call of Data Elements** window of the Logical View. In this window, you can ask for these checks to be or not to be generated in the Client component. For more details, see section List of Fields or Call of Elements.

  - ▪ the checks on the compatibility between fields

  These checks are managed with specific processing only.

  - ▪ the checks for the presence of Data Elements in a file or a database

  For example, to check that the town entered by the end user is in the

  database.

  As regards the check for Data Elements' presence, the value entered by the end user must be compared to the data contained in the database. This is standardly handled, but when needed, you can add specific processing.

Check

A check service is needed when you want to check only the data entered by the end user and to send him/her an error message if the value entered is incorrect. Logical View's data are not transferred to the database.

Implementation

To implement a check service, you must:

- declare the Logical View containing the service request issued by the Client component,
- perform the access to the external resources to check data consistency.

Reminder In the Dictionary, the description of external resources is specified on Segment type occurrences.

How to Declare the Logical View

The Business Component must know the **Logical View** for which it implements services.

You declare the Logical View in the Business Component's **Call of Segments (semi local)** window.

> The use of this window (menus, dialog boxes, etc.) is detailed in a specific chapter of the *VisualAge Pacbase WorkStation* User Interface Guide. In addition, complete documentation for all input fields is found in the *Pacbench C/S – Business Logic & TUI Clients Manual*.

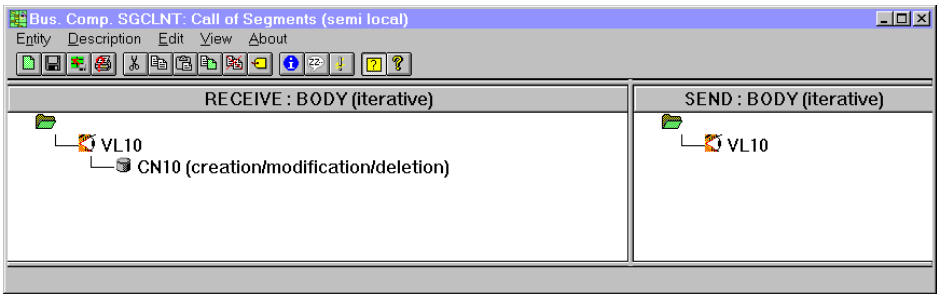
How to Access External Resources

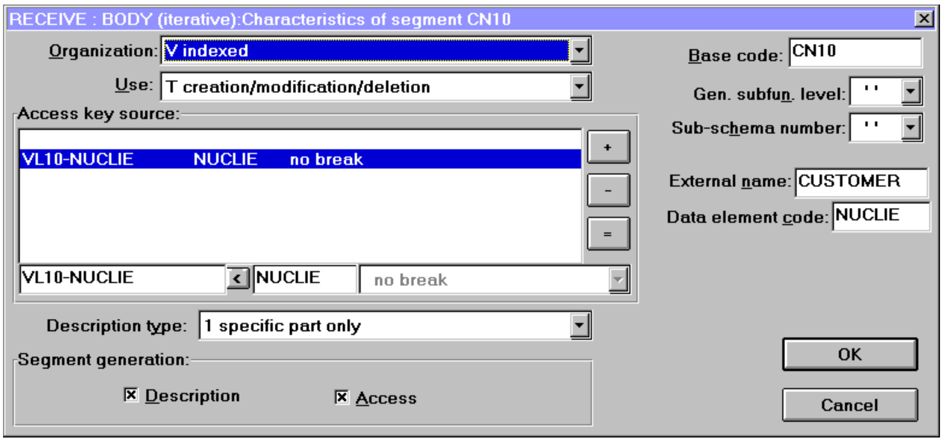
A check service may require access to one or more external resources.

The external resources are Segments representing a Table, a Record…. These Segments must be associated with the Logical View from which the data to be checked are sent.

To implement a check service, you specify the Segment code for the Business Component in the **Call of Segments (semi local)** window, in the **RECEIVE** panel. Indeed, the Business Component receives, from the Client component, the data to be checked via the Logical View.

The Segment used as reference for the check must be associated with the Logical View. You enter the Segment characteristics in the **Characteristics of segment** dialog box, opened by a double-click on the Segment code concerned.


Bus. Comp. SGCLNT: Call of Segments (semi local)
Entity Description Edit View About
RECEIVE : BODY (iterative)
VL10
CN10 (creation/modification/deletion)
SEND : BODY (iterative)
VL10



RECEIVE : BODY (iterative):Characteristics of segment CN10

Organization: V indexed

Use: T creation/modification/deletion

Access key source:

VL10-NUCLIE NUCLIE no break

VL10-NUCLIE NUCLIE no break

Description type: 1 specific part only

Segment generation:

☒ Description ☒ Access

Base code: CN10

Gen. subfun. level: ' '

Sub-schema number: ' '

External name: CUSTOMER

Data element code: NUCLIE

For a better functional understanding, we have grouped these characteristics into three thematic points (use, logical and physical characteristics).

To help you with input location, the characteristics to be entered in the Characteristics of Segment dialog box are marked with **[DB]**.

To implement the access to external resources, you declare the Segments required for the execution of the service and you indicate for each Segment:

- **The logical characteristics**

They make the connection with the Segment description in the VisualAge Pacbase Dictionary.

- the code of the Segment in the VisualAge Pacbase Dictionary **[DB]**
- the type of description **[DB]**:

blank by default, for a record with a possible common part (e.g. a Data Structure with only one Segment),

1 for a record without a common part (e.g. for databases, several Segments being defined in the same Data Structure),

2 for a Segment describing a remote database (ORACLE, SYBASE).

These characteristics are the same *for any type of requested service*: they are therefore declared only once for each Segment.

- The physical characteristics

They identify the physical data storage mode.

- the organization **[DB]:**

☞ For the W Organization:

Access generated via specific coding; only a description is generated in the Working-Storage Section.

- the external name **[DB]**:

DDNAME of the indexed file, or the VisualAge Pacbase code of the Database block in which the table is described, in case of SQL organization.

- the file physical key in KEY DATA ELEMENT **[DB]**:

This information comes under the physical characteristics only for certain organizations, among which the indexed files.

In this case you enter the code of the Data Element which must be specified to access a Segment (it may be a group Data Element).

These characteristics are the same *for any type of requested service*: they are therefore declared only once for each Segment.

- The **use characteristics** specific to a check / update service:

  They represent the nature of the data access and specify an access hierarchy.

  You indicate the type of Segment access required to perform the service as well as the sequence relative to the other Segments' accesses.

  Then the generator produces all the access functions via which the Business Component can perform the selection service for the Logical View.

The use characteristics are made up of the following information:

- the **access sequence** or **access hierarchy**. Access to a Segment is implemented for a given service on a Logical View. You must associate the Segment with the Logical View on which it depends.

  It may be necessary to access several Segments, you can:

  - associate each Segment with the Logical View:
    In the generated procedures, all check accesses are performed, whatever the results are,
  - or link the Segments to one another, thus making an access hierarchy:
    In the generated procedures, if a result of a check access is incorrect, the following accesses are not performed.

- the **category** by entering the Segment in one of the three following panels:

  INITIALIZATION

  BODY (iterative)

  For a multi-occurrence Logical View, this category performs an iterative access to the records.

  ☞ This category is **required** for a graphic application.

  TERMINATION

- the **access key source** **[DB]** contains the origin of the data which allows to specify the access key.
- the **access key** **[DB]** is the record key or the selection criterion.
- the **use type** **[DB]** for a selection service:

  E existence check only
  C consistency check and update by creation only,
  S consistency check and update by deletion only
  M consistency check and update by modification only,
  T consistency check and update by creation, modification or deletion,
  X consistency check and update by creation or modification.

Structure of the Generated Code

Working Storage Section

OPERT Area for the management of check/update/selection services

CH-view **view** represents the Logical View.

This area contains a table used for the checks of the Logical View.

It is generated if the CHECKSER=YES option has been specified in the Generation Parameters screen of the Dialogue (or Business Component).

This table contains one item for each Data Element called in the Logical View. The possible values (to be specified by the user in the Client component) are:

- N No check on this Data Element (default value)
- P Missing Data Element
- blank All Data Elements are checked

Linkage Section

TECH-ICATRC Number of instances to be processed by the Business Component on the Logical View.

TECH-IDATAC Check indicator of the Logical View, set by the Client component:

- blank Check on the fields of the Logical View whose check indicator contains a check request (see the description of the indicator in CONT-BUFFER).
- N No check
- C Check of all the Data Elements

ERR-BUFFER Error buffer, divided into errors on Segments and errors on Data Elements:

ERR-BUFSEG Segment access error buffer:

ERR-SEGCOD Code of the erroneous Segment

ERR-SEGERR Error code:

- DUPL Creation by mistake of a record, already existing record
- NFND Modification or deletion by mistake, non-existing record
- END End of list
- ABSC Record to be selected not found

ERR-SEGTYP Error type

ERR-ICATR Line number of the erroneous occurrence for a multi-occurrence processing

ERR-LIBRA Library code

ERR-SERVER Business Component code

ERR-DATA Buffer of errors on Data Element

ERR-LIBRA Library code

ERR-SERVER Business Component code

ERR-VIEW	Logical View code
ERR-DATCOD	Erroneous Data Element code
ERR-DATERR	Error code
ERR-DATTYP	Error type
ERR-ICATR	Line number of the erroneous occurrence for a multi-occurrence processing.
FILLER	End filler

Procedure Division

For the **'Logical View' block**, tags are as follows:

- the Logical View code in four characters:
  FVIEW
- the type of general processing, in four characters:
  CHUP check and/or update
- the category, in four characters:
  CATR multi-occurrence Logical View
- the service type, in four characters:
  SRVE check

All the services indicated in the table in Logical View Processing are generated but only this one will be actually executed if the Client requests a check service.

- the type of elementary procedure, in four characters:
  INIT initialization
  CHCK check
  DONE end

```
20     Check service             FVIEW-CHUP-CATR-SRVE
   25  Initializations           FVIEW-CHUP-CATR-SRVE-INIT
   25  Checks                    FVIEW-CHUP-CATR-SRVE-CHCK
   25  End of service            FVIEW-CHUP-CATR-SRVE-DONE
```

The **data transfer** function is not executed for this service but there may be a data check if CHECKSER=YES:

- the Logical View code in four characters: FVIEW
- data check, in four characters:
  CHKD check on the Logical View's Data Elements if CHECKSER=YES.

- the category, in four characters:
  CATR multi-occurrence Logical View

  if CHECKSER=YES:

```
10     Check on the View data               FVIEW-CHKD
 15    Multi-occurrence check               FVIEW-CHKD-CATR
   20  Check on Element DELCO1              FVIEW-CHKD-CATR-DELCO1
   20  Check on Element DELCO2              FVIEW-CHKD-CATR-DELCO2
```

....etc.

For **logical accesses** (=accesses to Segments required for the execution of the services on the Logical Views), tags are as follows:

- the Segment code in four characters:
  FSEGT
- the type of general processing, in four characters:
  CHCK check
- the category, in four characters:
  CATR multi-occurrence Logical View with a number of occurrences higher than 1
- the type of elementary processing, in four characters:
  ALIM key loading
  CALL call of the physical accesses (by **PERFORM**)
  ERRS error handling

```
10     Logical access to Segment for check     FSEGT-CHCK
 15    Multi-occurrence access                 FSEGT-CHCK-CATR
  20   Key loading                             FSEGT-CHCK-CATR-ALIM
  20   Call of physical access                 FSEGT-CHCK-CATR-CALL
  20   Error handling                          FSEGT-CHCK-CATR-ERRS
```

For **physical accesses**, tags are as follows:

- **F80**
- the Segment code in four characters:
  SEGT
- the access code:
  R read for check

```
10     Read for check                          F80-SEGT-R
```

Inserting Specific Code

VisualAge Pacbase allows you to implement processes that are not standard processes.

For example, you can specify a check for the compatibility between fields (ex: the date of purchase must be higher than that of availability). For the general rules for inserting specific code, refer to sections General Principles and Important Rules.

You first create a function where you identify the Logical View by its code. This approach is common to all the specific processes in the Logical View. This is why it is described in section Insertion Relative to the Logical View Processing.

Then, for each specific procedure to be inserted, you specify the procedure relative to a standard procedure of the Logical View. To do so, create a sub-function. Its code, title, hierarchical level and type of setting are common to all the specific procedures of the Logical View. This is why they are described in section Insertion Relative to the Logical View Processing.

Specific code can be inserted:

- In the Logical View processing (**FVIEW-CHUP** tag).

  - relative to the level 10 for check/update service. If you insert specific code at this level, it will be used for all the check/update services of the Logical View. It is the most generic specific code.

    The input is as follows:

    ```
    OPE OPERAND                            TY CONDITION
    Function XX   Sub-function YY      Level 10
     N   SUB-FCT TITLE                      type CHUP
    ```

    type represents the insertion type: *R to replace the standard processing, *A or *P to add a specific processing to the standard one. The level 10 is not mandatory for *A and *P types.

  - relative to the level 15 for check/update processing by category.

    The input is as follows:

    ```
    OPE OPERAND                          TY CONDITION
    Function XX   Sub-function YY      Level 15
     N   SUB-FCT TITLE                     type CHUP CATR
    ```

    Same comments as for the insertion at level 10.
    The CATR category is the only one possible for a check/update service.

  - relative to the level 20 for check/update processing by category and service type.

    The input is as follows:

    ```
    OPE OPERAND                          TY CONDITION
    Function XX   Sub-function YY    Level 20
     N   SUB-FCT TITLE                 type CHUP CATR SRVE
    ```

    Same comments as for the previous insertion.
    SRVE represents the check service.

- In the logical accesses (FSEGT-CHCK tag):

  - relative to the level 10 for the access to a Segment for data check:

    The input is as follows:

    ```
    OPE OPERAND                          TY CONDITION
    Function XX   Sub-function YY   Level 10
     N   SUB-FCT TITLE                     type CHCK segt
    ```

    type represents the insertion type: *R to replace the standard processing, *A or *P to add a specific processing to the standard one. The level 10 is not mandatory for *A and *P types.

    segt represents the Segment code.

  - relative to the level 15 for the access to a Segment for data check by category.

    The input is as follows:

    ```
    OPE OPERAND                          TY CONDITION
    Function XX   Sub-function YY   Level 15
     N   SUB-FCT TITLE                 type CHCK segt CATR
    ```

    Same comments as for the insertion at level 10.
    The CATR category is the only one possible for a check/update service.

  - relative to the level 20 for the access to a Segment for data check by category and elementary service type.

    The input is as follows:

    ```
    OPE OPERAND                         TY CONDITION
    Function XX   Sub-function YY   Level 20
     N   SUB-FCT TITLE                    type CHCK segt CATR proc
    ```

    Same comments as for the previous insertion. proc represents the elementary processing type (ALIM for key loading, CALL for the call of physical processing, ERRS for error processing).

- In the physical accesses (**F80-SEGT-R** tag).

Refer to section Insertion Relative to the 'Physical Accesses' Level.

Read

A read service is needed when you want to perform a direct read in the database (and not a sequential read as that is the case with a selection service). This service can be used to perform an additional read before an update (for example, to display the product label corresponding to the product code entered). This service sends no error if the read is unsuccessful and just displays an information message.

Implementation

To implement a read service, you must:

- declare the Logical View containing the service request issued by the Client component,
- perform the access to the external resources to find the data to be read,
- implement the data transfer from the Logical View to the Segments.

Reminder In the Dictionary, the description of external resources is specified on Segment type occurrences.

How to Declare the Logical View

The Business Component must know the **Logical View** for which it implements services.

You declare the Logical View in the Business Component's **Call of Segments (semi local)** window.

> The use of this window (menus, dialog boxes, etc.) is detailed in a specific chapter of the *VisualAge Pacbase WorkStation* User Interface Guide. In addition, complete documentation for all input fields is found in the *Pacbench C/S – Business Logic & TUI Clients Manual*.

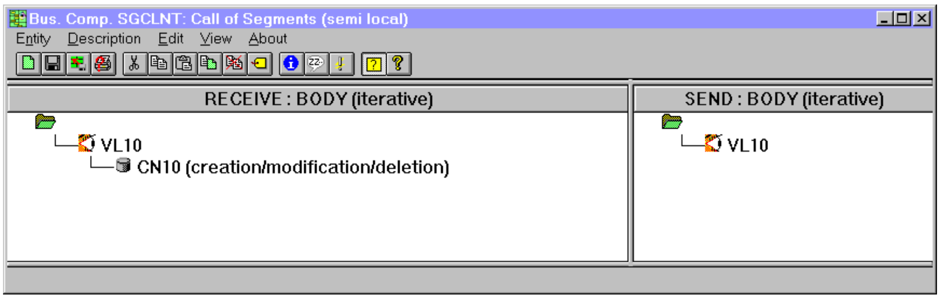
How to Access External Resources

A read service may require access to several external resources.

The external resources are Segments representing a Table, a Record…. These Segments must be associated with the Logical View which indicates what data must be read.

To implement a read service, you specify the Segment code in the **Call of Segments (semi local)** window, in the **RECEIVE** panel only. Indeed, the Business Component receives from the Client component, the keys allowing to access the data to be read via the Logical View.

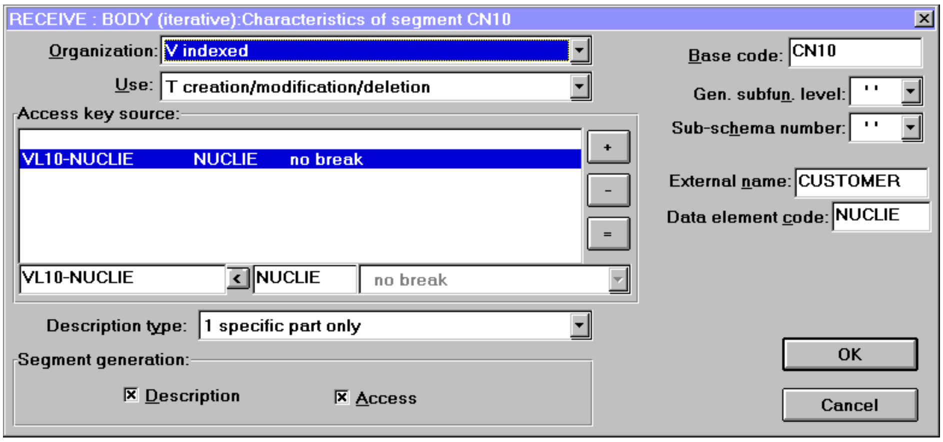
The Segment used as reference for the read must be associated with the Logical View. You enter the characteristics of the Segment in the `Characteristics of Segment` dialog box that you open by double-clicking on the Segment code.



Bus. Comp. SGCLNT: Call of Segments (semi local)

RECEIVE : BODY (iterative)
- VL10
  - CN10 (creation/modification/deletion)

SEND : BODY (iterative)
- VL10



RECEIVE : BODY (iterative):Characteristics of segment CN10

Organization: V indexed

Use: T creation/modification/deletion

Access key source:

VL10-NUCLIE NUCLIE no break

VL10-NUCLIE NUCLIE no break

Description type: 1 specific part only

Segment generation:

☒ Description ☒ Access

Base code: CN10

Gen. subfun. level: ' '

Sub-schema number: ' '

External name: CUSTOMER

Data element code: NUCLIE

For a better functional understanding, we have grouped these characteristics into three thematic points (use, logical and physical characteristics).

To help you with input location, the characteristics to be entered in the Characteristics of Segment dialog box are marked with **[DB]**.

To implement the access to external resources, you declare the Segments required for the execution of the service and you indicate for each Segment:

- **The logical characteristics**

They make the connection with the Segment description in the VisualAge Pacbase Dictionary.

- the code of the Segment in the VisualAge Pacbase Dictionary **[DB]**
- the type of description **[DB]**:

`blank` by default, for a record with a possible common part (e.g. a Data Structure with only one Segment),

`1` for a record without a common part (e.g. for databases, several Segments being defined in the same Data Structure),

`2` for a Segment describing a remote database (ORACLE, SYBASE).

These characteristics are the same *for any type of requested service*: they are therefore declared only once for each Segment.

- The physical characteristics

They identify the physical data storage mode.

- the organization **[DB]**:
  - ☞ For the **W** Organization:

    Access generated via specific coding; only a description is generated in the Working-Storage Section.
- the external name **[DB]**:

  DDNAME of the indexed file, or the VisualAge Pacbase code of the Database block in which the table is described, in case of SQL organization.
- the file physical key in **KEY DATA ELEMENT** **[DB]**:

  This information comes under the physical characteristics only for certain organizations, among which the indexed files.

  In this case you enter the code of the Data Element which must be specified to access a Segment (it may be a group Data Element).

  These characteristics are the same *for any type of requested service*: they are therefore declared only once for each Segment.

- The **use characteristics** specific to a check / update service:

  They represent the nature of the data access and specify an access hierarchy.

  You indicate the type of Segment access required to perform the service as well as the sequence relative to the other Segments' accesses.

  Then the generator produces all the access functions via which the Business Component can perform the selection service for the Logical View.

The use characteristics are made up of the following information:

- the **access sequence** or **access hierarchy**. Access to a Segment is implemented for a given service on a Logical View. You must associate the Segment with the Logical View on which it depends.

  It may be necessary to access several Segments, you can:
  - ♦ associate each Segment with the Logical View:
    In the generated procedures, all check accesses are performed, whatever the results are,
  - ♦ or link the Segments to one another, thus making an access hierarchy:
    In the generated procedures, if a result of a check access is incorrect, the following accesses are not performed.
- the **category** by entering the Segment in one of the three following panels:

  INITIALIZATION

  BODY (iterative)

  For a multi-occurrence Logical View, this category performs an iterative access to the records.

  ☞ This category is **required** for a graphic application.

  TERMINATION
- the **access key source** **[DB]** contains the origin of the data which allows to specify the access key.
- the **access key** **[DB]** is the record key or the selection criterion.

- The use characteristics:
  - the use type [DB]:

    L read only,
    C consistency check and update by creation only,
    M consistency check and update by modification only,
    S consistency check and update by deletion only
    T consistency check and update by creation, modification or deletion,
    X consistency check and update by creation or modification.

How to Transfer Data

The Client component's data from which the read is performed is not transferred to the Logical View. Conversely, the data retrieved in the Segments are transferred to the Logical View to be processed.

For details, see section Data Transfer.

Structure of the Generated Code

Working Storage Section

OPERT Area for the management of check/update/selection services

CH-view **view** represents the Logical View.

This area contains a table used for the checks of the Logical View.

It is generated if the CHECKSER=YES option has been specified in the Generation Parameters screen of the Dialogue (or Business Component).

This table contains one item for each Data Element called in the Logical View. The possible values (to be specified by the user in the Client component) are:

N No check on this Data Element (default value)
P Missing Data Element
blank All Data Elements are checked

Linkage Section

TECH-ICATRC Number of instances to be processed by the Business Component on the Logical View.

TECH-IDATAC Check indicator of the Logical View, set by the Client component:

blank Check on the fields of the Logical View whose check indicator contains a check request (see the description of the indicator in CONT-BUFFER).

N No check
C Check on all the Logical View's fields

ERR-BUFFER Error buffer, divided into errors on Segments and errors on Data Elements:

ERR-BUFSEG Segment access error buffer:

ERR-SEGCOD	Code of the erroneous Segment
ERR-SEGERR	Error code:
DUPL	Creation by mistake of a record, already existing record
NFND	Modification or deletion by mistake, non-existing record
END	End of list
ABSC	Record to be selected not found
ERR-SEGTYP	Error type
ERR-ICATR	Line number of the erroneous occurrence for a multi-occurrence processing
ERR-LIBRA	Library code
ERR-SERVER	Business Component code
ERR-DATA	Buffer of errors on Data Element
ERR-LIBRA	Library code
ERR-SERVER	Business Component code
ERR-VIEW	Logical View code
ERR-DATCOD	Erroneous Data Element code
ERR-DATERR	Error code
ERR-DATTYP	Error type
ERR-ICATR	Line number of the erroneous occurrence for a multi-occurrence processing.
FILLER	End filler

Procedure Division

For the **'Logical View' block**, tags are as follows:

- the Logical View code in four characters:
  FVIEW
- the type of general processing, in four characters:
  CHUP check and/or update
- the category, in four characters:
  CATR multi-occurrence Logical View
- the service type, in four characters:
  SRVL read

(All the services indicated in the table in section Logical View Processing are generated but only this one will be actually executed if the Client requests a read service).

- the type of elementary procedure, in four characters:
  INIT initialization
  CHCK check
  TRAN transfer
  DONE end

```
20  Read service                FVIEW-CHUP-CATR-SRVL
 25 Initializations             FVIEW-CHUP-CATR-SRVL-INIT
 25 Checks                      FVIEW-CHUP-CATR-SRVL-CHCK
 25 Transfers                   FVIEW-CHUP-CATR-SRVL-TRAN
 25 End of service              FVIEW-CHUP-CATR-SRVL-DONE
```

The read service performs **data transfers** from the Logical View to the Segment. Tags are as follows:

- the Logical View code in four characters:
  FVIEW
- the transfer direction, in four characters:
  TRVW TRansfer to the "VieW" in the direction Segment → View
  CHKD check on the Logical View's Data Elements if CHECKSER=YES.
- the category, in four characters:
  CATR multi-occurrence Logical View

```
10     Data transfers to View                  FVIEW-TRVW
 15    Multi-occurrence transfers              FVIEW-TRVW-CATR
       if CHECKSER=YES:
10     Checks on View data                     FVIEW-CHKD
 15    Multi-occurrence checks                 FVIEW-CHKD-CATR
  20   Check on Element DELCO1                 FVIEW-CHKD-CATR-DELCO1
  20   Check on Element DELCO2                 FVIEW-CHKD-CATR-DELCO2
```

For **logical accesses** (=accesses to Segments required for the execution of the services on the Logical Views), tags are as follows:

- the Segment code in four characters:
  FSEGT
- the type of general processing, in four characters:
  SLCT selection
- the category, in four characters:
  CATR multi-occurrence Logical View with a number of occurrences higher than 1
- the type of elementary processing, in four characters:
  ALIM key loading
  CALL call of the physical accesses (by PERFORM)
  ERRS error handling

```
10     Logical access to Segment for slct      FSEGT-SLCT
 15    Multi-occurrence access                 FSEGT-SLCT-CATR
  20   Key loading                             FSEGT-SLCT-CATR-ALIM
  20   Call of physical access                 FSEGT-SLCT-CATR-CALL
  20   Error handling                          FSEGT-SLCT-CATR-ERRS
```

For **physical accesses**, tags are as follows:

- F80
- the Segment code in four characters: SEGT
- the access code:
  R read for check

```
10     Read for check            F80-SEGT-R
```

Inserting Specific Code

VisualAge Pacbase allows you to implement processes that are not standard processes.

For example, before executing an update, you can request the display of the actual unit price of an item from a order line taking into account the granted discount. To do so, the standard unit price stored in the database must be retrieved and, according to the number and amount indicated in the order line, the actual unit price is computed.

For the general rules for inserting specific code, refer to sections General Principles and Important Rules.

You first create a function where you identify the Logical View by its code. This approach is common to all the specific processes in the Logical View. This is why it is described in section Insertion Relative to the Logical View Processing.

Then, for each specific procedure to be inserted, you specify the procedure relative to a standard procedure of the Logical View. To do so, create a sub-function. Its code, title, hierarchical level and type of setting are common to all the specific procedures of the Logical View. This is why they are described in section Insertion Relative to the Logical View Processing.

Specific code can be inserted:

- In the Logical View processing (**FVIEW-CHUP** tag).
  - relative to the level **10** for check/update service. If you insert specific code at this level, it will be used for all the check/update services of the Logical View. It is the most generic specific code.

    The input is as follows:
    ```
    OPE OPERAND                            TY CONDITION
    Function XX   Sub-function YY      Level 10
     N   SUB-FCT TITLE                       type CHUP
    ```
    type represents the insertion type: ***R** to replace the standard processing, ***A** or ***P** to add a specific processing to the standard one. The level **10** is not mandatory for ***A** and ***P** types.
  - relative to the level **15** for check/update processing by category.

    The input is as follows:
    ```
    OPE OPERAND                          TY CONDITION
    Function XX   Sub-function YY      Level 15
     N   SUB-FCT TITLE                     type CHUP CATR
    ```
    Same comments as for the insertion at level **10**.
    The **CATR** category is the only one possible for a check/update service.

  - relative to the level **20** for check/update processing by category and service type.

  The input is as follows:
    ```
    OPE OPERAND                          TY CONDITION
    Function XX   Sub-function YY   Level 20
     N   SUB-FCT TITLE                 type CHUP CATR SRVL
    ```
    Same comments as for the previous insertion. **SRVL** represents the read service.

- In the data transfers from the Segment to the Logical View (**FVIEW-TRVW** tag):

  Relative to level **10** for the transfer direction:

    The input is as follows:
    ```
    OPE OPERAND                          TY CONDITION
    Function XX   Sub-function YY   Level 10
    ```

```
 N   SUB-FCT TITLE                         type TRVW
```

`type` represents the insertion type: `*R` to replace the standard processing, `*A` or `*P` to add a specific processing to the standard one. The level `10` is not mandatory for `*A` and `*P` types.

- relative to the level `15` for the transfer direction by category.

  The input is as follows:

  ```
  OPE OPERAND                           TY CONDITION
  Function XX   Sub-Function YY   Level 15
   N SUB-FCT TITLE                        type TRVW CATR
  ```

  Same comments as for the insertion at level `10`. The `CATR` category is the only one possible.

- In the logical accesses (`FSEGT-SLCT` tag):
  - relative to the level `10` for the access to a Segment for data selection:

    The input is as follows:

    ```
    OPE OPERAND                           TY CONDITION
    Function XX   Sub-function YY     Level 10
     N SUB-FCT TITLE                        type SLCT segt
    ```

    `type` represents the insertion type: `*R` to replace the standard processing, `*A` or `*P` to add a specific processing to the standard one. The level `10` is not mandatory for `*A` and `*P` types. `segt` represents the Segment code.

  - relative to the level `15` for the access to a Segment for data selection by category.

    The input is as follows:

    ```
    OPE OPERAND                           TY CONDITION
    Function XX   Sub-function YY     Level 15
     N SUB-FCT TITLE                     type SLCT segt CATR
    ```

    Same comments as for the insertion at level `10`.
    `CATR` is the only possible category.

  - relative to the level `20` for the access to a Segment for data selection by category and elementary service type.

    The input is as follows:

    ```
    OPE OPERAND                         TY CONDITION
    Function XX   Sub-function YY     Level 20
     N SUB-FCT TITLE                  type SLCT segt CATR proc
    ```

    Same comments as for the previous insertion. `proc` represents the elementary processing type (`ALIM` for key loading, `CALL` for the call of physical processing, `ERRS` for error processing).

- In the physical accesses (`F80-SEGT-R` tag).

Refer to section Insertion Relative to the ‘Physical Accesses’ Level.

Check and Update

A check and update service is needed when you want to check the data entered by the end user (if necessary, with the display of an error message) and update the database.

Implementation

To implement a check and update service, you must:

- declare the Logical View containing the service request issued by the Client component,
- perform the access to the external resources to check data consistency.
- implement the data transfer from the Logical View to the data of the record to be updated,
- update the data in the database.

Reminder In the Dictionary, the description of external resources is specified on Segment type occurrences.

How to Declare the Logical View

The Business Component must know the **Logical View** for which it implements services.

You declare the Logical View in the Business Component's **`Call of Segments (semi local)`** window.

The use of this window (menus, dialog boxes, etc.) is detailed in a specific chapter of the *VisualAge Pacbase WorkStation* User Interface Guide. In addition, complete documentation for all input fields is found in the *Pacbench C/S – Business Logic & TUI Clients Manual*.

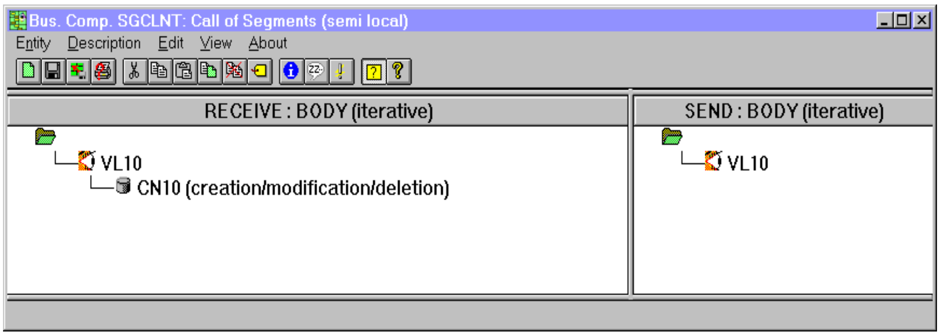
How to Access External Resources

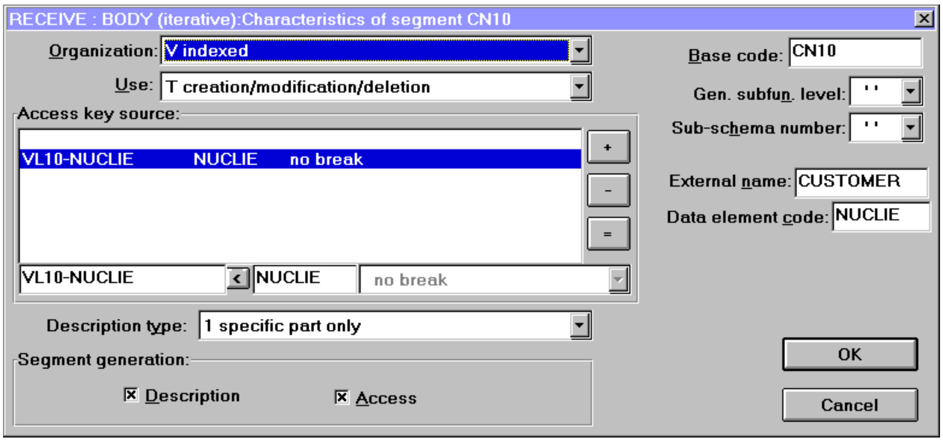
A check and update service may require access to several external resources, in particular to perform integrity checks of the external resource to be updated against other external resources.

The external resources are Segments representing a Table, a Record, etc. These Segments must be associated with the Logical View which indicates what data must be checked.

To implement a check and update service, you specify the Segment code in the **`Call of Segments (semi local)`** window, in the **`RECEIVE`** panel only. Indeed, the Business Component receives from the Client component, the data to be checked and updated via the Logical View.

The Segment used as reference for the check must be associated with the Logical View. You enter the characteristics of the Segment in the **`Characteristics of segment`** dialog box that you open by double-clicking on the Segment code.


Bus. Comp. SGCLNT: Call of Segments (semi local)
Entity
Description
Edit
View
About
RECEIVE : BODY (iterative)
SEND : BODY (iterative)
VL10
CN10 (creation/modification/deletion)
VL10



RECEIVE : BODY (iterative):Characteristics of segment CN10

Organization: V indexed

Use: T creation/modification/deletion

Access key source:

VL10-NUCLIE NUCLIE no break

VL10-NUCLIE NUCLIE no break

Base code: CN10

Gen. subfun. level: ' '

Sub-schema number: ' '

External name: CUSTOMER

Data element code: NUCLIE

Description type: 1 specific part only

Segment generation: ☒ Description ☒ Access

For a better functional understanding, we have grouped these characteristics into three thematic points (use, logical and physical characteristics).

To help you with input location, the characteristics to be entered in the Characteristics of Segment dialog box are marked with **[DB]**.

To implement the access to external resources, you declare the Segments required for the execution of the service and you indicate for each Segment:

- **The logical characteristics**

They make the connection with the Segment description in the VisualAge Pacbase Dictionary.

  - the code of the Segment in the VisualAge Pacbase Dictionary **[DB]**
  - the type of description **[DB]**:

    `blank` by default, for a record with a possible common part (e.g. a Data Structure with only one Segment),

    `1` for a record without a common part (e.g. for databases, several Segments being defined in the same Data Structure),

    `2` for a Segment describing a remote database (ORACLE, SYBASE).

These characteristics are the same *for any type of requested service*: they are therefore declared only once for each Segment.

- The physical characteristics

They identify the physical data storage mode.

  - the organization **[DB]**:

    ☞ For the `W` Organization:

    Access generated via specific coding; only a description is generated in the Working-Storage Section.

  - the external name **[DB]**:

    `DDNAME` of the indexed file, or the VisualAge Pacbase code of the Database block in which the table is described, in case of SQL organization.

  - the file physical key in `KEY DATA ELEMENT` **[DB]**:

This information comes under the physical characteristics only for certain organizations, among which the indexed files.

In this case you enter the code of the Data Element which must be specified to access a Segment (it may be a group Data Element).

These characteristics are the same *for any type of requested service*: they are therefore declared only once for each Segment.

- The **use characteristics** specific to a check / update service:

  They represent the nature of the data access and specify an access hierarchy.

  You indicate the type of Segment access required to perform the service as well as the sequence relative to the other Segments' accesses.

  Then the generator produces all the access functions via which the Business Component can perform the selection service for the Logical View.

The use characteristics are made up of the following information:

- the **access sequence** or **access hierarchy**. Access to a Segment is implemented for a given service on a Logical View. You must associate the Segment with the Logical View on which it depends.

  It may be necessary to access several Segments, you can:

  - associate each Segment with the Logical View:
    In the generated procedures, all check accesses are performed, whatever the results are,
  - or link the Segments to one another, thus making an access hierarchy:
    In the generated procedures, if a result of a check access is incorrect, the following accesses are not performed.

- the **category** by entering the Segment in one of the three following panels:

  INITIALIZATION

  BODY (iterative)

  For a multi-occurrence Logical View, this category performs an iterative access to the records.

  ☞ This category is **required** for a graphic application.

  TERMINATION

- the **access key source [DB]** contains the origin of the data which allows to specify the access key.
- the **access key [DB]** is the record key or the selection criterion.

- The use characteristics:
  - the use type [DB]:

    C consistency check and update by creation only,
    M consistency check and update by modification only,
    S consistency check and update by deletion only
    T consistency check and update by creation, modification or deletion,
    X consistency check and update by creation or modification.

How to Transfer Data

To implement a check and update service, after the access to the external resources for check and before the access for update, the data transfer is carried out from the Logical View's data to the Segments.

For details, refer to section Data Transfer.

How to Update Data

The update is automatically carried out if the Segment is assigned with one of the use types previously mentioned and if the Business Component is called for an update service by the Client component.

The update service sends to the storage area (database, file, etc) the data that has been updated in the Client component.

Structure of the Generated Code

Working Storage Section

`CATM` Transaction code:

`C` Creation

`M` Modification

`A` Deletion

`X` Implicit update

`OPERT` Area for the management of check/update/selection services

`A-CATM` Description buffer. This buffer contains the transaction codes related to both non repeated and repeated data of the Logical View.

`A-CATM-CA` Action code applying to non repeated data. Systematically generated.

`A-CATM-CR OCCURS N`

Action code applying to repeated data.

N is the maximum number of repetitions for all the Logical Views in this server. Generated if one of the Business Component's Logical View contains repeated data.

CH-view **view** represents the Logical View.

This area contains a table used for the checks of the Logical View.

It is generated if the **`CHECKSER=YES`** option has been specified in the Generation Parameters screen of the Dialogue (or Business Component).

This table contains one item for each Data Element called in the Logical View. The possible values (to be specified by the user in the Client component) are:

`N` No check on this Data Element (default value)

`P` Missing Data Element

`blank` All Data Elements are checked

Linkage Section

TECH-ICATRC Number of instances to be processed by the Business Component on the Logical View.

TECH-IDATAC Check indicator of the Logical View data, set by the Client component:

blank Check on the fields of the Logical View whose check indicator contains a check request (see the description of the indicator in CONT-BUFFER).
N No check
C Check on all the Logical View's fields

ERR-BUFFER Error buffer, divided into errors on Segments and errors on Data Elements:

ERR-BUFSEG Segment access error buffer:

ERR-SEGCOD Code of the erroneous Segment

ERR-SEGERR Error code:
DUPL Creation by mistake of a record, already existing record
NFND Modification or deletion by mistake, non-existing record
END End of list
ABSC Record to be selected not found

ERR-SEGTYP Error type

ERR-ICATR Line number of the erroneous occurrence for a multi-occurrence processing

ERR-LIBRA Library code

ERR-SERVER Business Component code

ERR-DATA Buffer of errors on Data Element

ERR-LIBRA Library code

ERR-SERVER Business Component code

ERR-VIEW Logical View code

ERR-DATCOD Erroneous Data Element code

ERR-DATERR Error code

ERR-DATTYP Error type

ERR-ICATR Line number of the erroneous occurrence for a multi-occurrence processing.

FILLER End filler

Procedure Division

For the **'Logical View' block**, tags are as follows:

- the Logical View code in four characters:
  FVIEW
- the type of general processing, in four characters:
  CHUP check and/or update
- the category, in four characters:
  CATR multi-occurrence Logical View

- the service type, in four characters:

SRVM	check and update

(All the services indicated in the table in section Logical View Processing are generated but only this one will be actually executed if the Client requests a check and update service).

- the type of elementary procedure, in four characters:

INIT	initialization
CHCK	check
TRAN	transfer
UPDT	update
DONE	end

```
20  chck/updt service          FVIEW-CHUP-CATR-SRVM
 25 Initializations            FVIEW-CHUP-CATR-SRVM-INIT
 25 Checks                     FVIEW-CHUP-CATR-SRVM-CHCK
 25 Transfers                  FVIEW-CHUP-CATR-SRVM-TRAN
 25 Update                     FVIEW-CHUP-CATR-SRVM-UPDT
 25 End of service             FVIEW-CHUP-CATR-SRVM-DONE
```

The check and update service performs **data transfers** from the Logical View to the Segment. Tags are as follows:

- the Logical View code in four characters:
  FVIEW
- the transfer direction, in four characters:

TRDT	TRansfer to the "DaTa" in the direction View → Segment
CHKD	check on the Logical View's Data Elements If CHECKSER=YES.

- the category, in four characters:

CATR	multi-occurrence Logical View

```
  10      Data transfers to Segment        FVIEW-TRDT
   15     Multi-occurrence transfers       FVIEW-TRDT-CATR
```

if CHECKSER=YES:

```
  10      Checks on View data              FVIEW-CHKD
   15     Multi-occurrence checks          FVIEW-CHKD-CATR
   20     Check on Element DELCO1          FVIEW-CHKD-CATR-DELCO1
   20     Check on Element DELCO2          FVIEW-CHKD-CATR-DELCO2
```

For **logical accesses** (=accesses to Segments required for the execution of the services on the Logical Views), tags are as follows:

- the Segment code in four characters:
  FSEGT
- the type of general processing, in four characters:

CHCK	check
UPDT	update

- the category, in four characters:

CATR	multi-occurrence Logical View with a number of occurrences higher than 1

- the type of elementary processing, in four characters:

ALIM	key loading

CALL	call of the physical accesses (by **PERFORM**)
ERRS	error handling

```
10     Logical access to Segment for check     FSEGT-CHCK
 15    Multi-occurrence access                 FSEGT-CHCK-CATR
  20   Key loading                             FSEGT-CHCK-CATR-ALIM
  20   Call of physical access                 FSEGT-CHCK-CATR-CALL
  20   Error handling                          FSEGT-CHCK-CATR-ERRS
10     Logical access to Segment for updt      FSEGT-UPDT
 15    Multi-occurrence access                 FSEGT-UPDT-CATR
  20   Key loading                             FSEGT-UPDT-CATR-ALIM
  20   Call of physical access                 FSEGT-UPDT-CATR-CALL
  20   Error handling                          FSEGT-UPDT-CATR-ERRS
```

For **physical accesses**, tags are as follows:

- **F80**
- the Segment code in four characters: **SEGT**
- the access code:

R	Read for check
RU	Read for check before update
W	Creation (write)
RW	Modification (rewrite)
D	Deletion (Delete)
UN	Unlock of record read in RU

```
10       Read for check                      F80-SEGT-R
10       Read for check before update        F80-SEGT-RU
10       Creation (write)                    F80-SEGT-W
10       Modification (rewrite)              F80-SEGT-RW
10       Deletion (delete)                   F80-SEGT-D
10       Unlock of record read in RU         F80-SEGT-UN
```

Inserting Specific Code

VisualAge Pacbase allows you to implement processes that are not standard processes.

For example, before the update, you can compute the unit price charged for an item in an order line, taking into account the granted discount, to store it in the database (without displaying it in the end user interface). To do so, retrieve the amount specified in the order line and the number of items specified in the application to compute the price.

For the general rules for inserting specific code, refer to sections General Principles and Important Rules.

You first create a function where you identify the Logical View by its code. This approach is common to all the specific processes in the Logical View. This is why it is described in section Insertion Relative to the Logical View Processing.

Then, for each specific procedure to be inserted, you specify the procedure relative to a standard procedure of the Logical View. To do so, create a sub-function. Its code, title, hierarchical level and type of setting are common to all the specific procedures of the Logical View. This is why they are described in section Insertion Relative to the Logical View Processing.

Specific code can be inserted:

- In the Logical View processing (**FVIEW-CHUP** tag).

- relative to the level `10` for check/update service. If you insert specific code at this level, it will be used for all the check/update services of the Logical View. It is the most generic specific code.

  The input is as follows:

  ```
  OPE OPERAND                            TY CONDITION
  Function XX   Sub-function YY      Level 10
   N   SUB-FCT TITLE                      type CHUP
  ```

  `type` represents the insertion type: `*R` to replace the standard processing, `*A` or `*P` to add a specific processing to the standard one. The level `10` is not mandatory for `*A` and `*P` types.

- relative to the level `15` for check/update processing by category.

  The input is as follows:

  ```
  OPE OPERAND                          TY CONDITION
  Function XX   Sub-function YY      Level 15
   N   SUB-FCT TITLE                    type CHUP CATR
  ```

  Same comments as for the insertion at level `10`.
  The `CATR` category is the only one possible for a check/update service.

- relative to the level `20` for the access to a Segment for check/update processing by category and elementary service type.

The input is as follows:

```
OPE OPERAND                          TY CONDITION
Function XX   Sub-function   YY    Level 20
 N SUB-FCT TITLE                   type CHUP CATR SRVM
```

Same comments as for the previous insertion. `SRVM` represents the check/update service.

- In the **data transfers** from the Logical View to the Segment (`FVIEW-TRDT` tag):
  - Relative to level `10` for the transfer direction:

    The input is as follows:

    ```
    OPE OPERAND                          TY CONDITION
    Function XX   Sub-function    YY    Level 10
     N SUB-FCT TITLE                      type TRDT
    ```

    `type` represents the insertion type: `*R` to replace the standard processing, `*A` or `*P` to add a specific processing to the standard one. The level `10` is not mandatory for `*A` and `*P` types.

  - relative to the level `15` for the check/update by category.

    The input is as follows:

    ```
    OPE OPERAND                          TY CONDITION
    Function XX   Sub-function YY   Level 15
     N SUB-FCT TITLE                       type TRDT CATR
    ```

    Same comments as for the insertion at level `10`.
    The `CATR` category is the only one possible.

- In the **logical accesses** (`FSEGT-CHCK` and `FSEGT-UPDT` tags):
  - relative to the level `10` for the access to a Segment for data check and update:

    The input is as follows:

    ```
    OPE OPERAND                          TY CONDITION
    Function XX   Sub-function YY   Level 10
     N SUB-FCT TITLE                       type genp segt
    ```

type represents the insertion type: ***R** to replace the standard processing, ***A** or ***P** to add a specific processing to the standard one. The level **10** is not mandatory for ***A** and ***P** types. **genp** represents the type of general processing: **CHCK** or **UPDT**. **segt** represents the Segment code.

- relative to the level **15** for the access to a Segment for data check and update by category.

  The input is as follows:

  ```
  OPE OPERAND                          TY CONDITION
  Function XX   Sub-function YY   Level 15
   N SUB-FCT TITLE                   type genp segt CATR
  ```

  Same comments as for the insertion at level **10**.

  CATR is the only possible category.

- relative to the level **20** for the access to a Segment for data check and update by category and elementary service type.

  The input is as follows:

  ```
  OPE OPERAND                          TY CONDITION
  Function XX   Sub-function YY   Level 20
   N SUB-FCT TITLE                 type genp segt CATR elmp
  ```

  Same comments as for the previous insertion.

  elmp represents the elementary processing type (**ALIM** for key loading, **CALL** for the call of physical processing, **ERRS** for error processing).

- In the **physical accesses** (**F80-SEGT-R, F80-SEGT-RU, F80-SEGT-W, F80-SEGT-RW, F80-SEGT-D, F80-SEGT-UN** tags).

Refer to section Insertion Relative to the ‘Physical Accesses’ Level.

Check / Update / Selection

A check/update/selection service is needed when you want to check the data entered by the user (if necessary, with the display of an error message), display the information in the Client component according to a specified format and refresh this information with the user input, while updating the database.

Implementation

To implement a check/update/selection service, you must:

- declare the Logical View containing the service request issued by the Client component,
- perform the access to the external resources to check data consistency and select the data to be sent to the Client component,
- implement the data transfer from the Logical View to the data of the record to be updated,
- update the data in the database,
- perform the selection in the Client while taking the update into account.

Reminder In the Dictionary, the description of external resources is specified on Segment type occurrences.

How to Declare the Logical View

The Business Component must know the **Logical View** for which it implements services.

You declare the Logical View in the Business Component's `Call of Segments (semi local)` window.

> The use of this window (menus, dialog boxes, etc.) is detailed in a specific chapter of the *VisualAge Pacbase WorkStation* User Interface Guide. In addition, complete documentation for all input fields is found in the *Pacbench C/S – Business Logic & TUI Clients Manual*.

How to Access External Resources

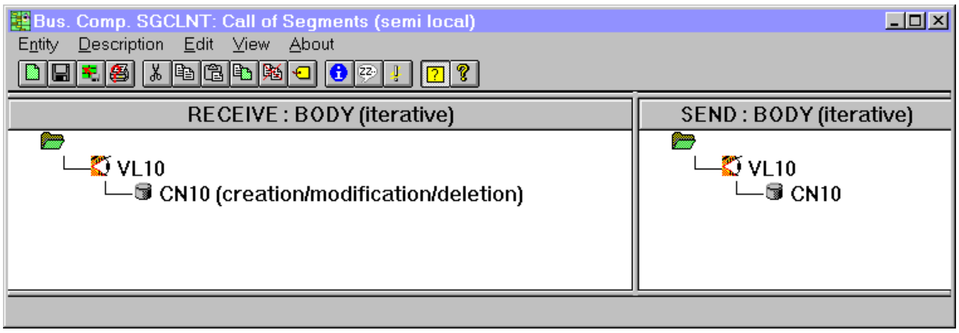
A check/update/selection service may require access to several external resources.

To implement a check/update/selection service, you specify the Segment code in the `Call of Segments (semi local)` window:

- in the `RECEIVE` panel for the Business Component receives from the Client component, the data to be checked and updated via the Logical View,
- and in the `SEND` panel for the Business Component sends the data to the Client component to refresh its information.

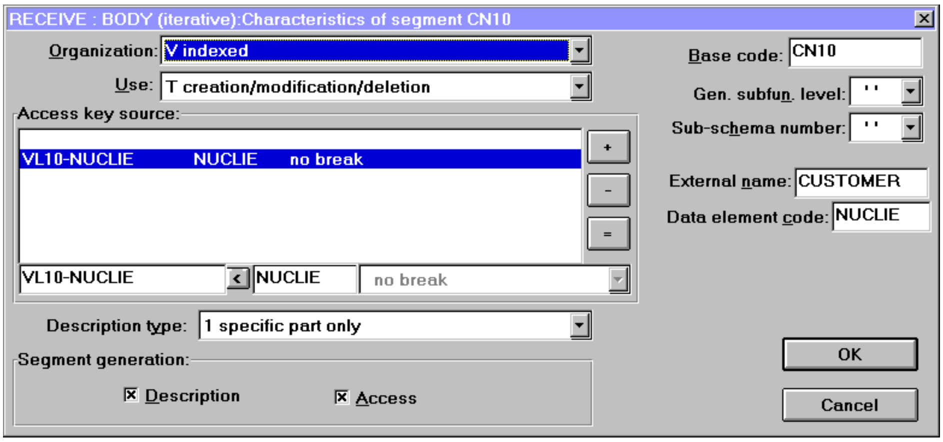
If no Segment is entered in the `SEND` panel, the selection service will not be executed and only a direct read will be performed.

The Segment(s) used as reference(s) for check and update can be different from that(those) used as reference(s) for selection but all of them must be associated with the same Logical View.


Bus. Comp. SGCLNT: Call of Segments (semi local)
Entity
Description
Edit
View
About
RECEIVE : BODY (iterative)
VL10
CN10 (creation/modification/deletion)
SEND : BODY (iterative)
VL10
CN10

The following comments apply to the access to the external resources for the check/update service (i.e to the Segment indicated in the `RECEIVE` panel). For the specification of the access to the external resources for the selection service, refer to sectionHow to Access External Resources.

You specify the characteristics of the Segment used as reference for check/update in the `Characteristics of Segment` dialog box, that you open by double-clicking on the Segment code.



RECEIVE : BODY (iterative):Characteristics of segment CN10

Organization: V indexed

Use: T creation/modification/deletion

Access key source:

VL10-NUCLIE NUCLIE no break

VL10-NUCLIE NUCLIE no break

Base code: CN10

Gen. subfun. level: ' '

Sub-schema number: ' '

External name: CUSTOMER

Data element code: NUCLIE

Description type: 1 specific part only

Segment generation:

☒ Description ☒ Access

For a better functional understanding, we have grouped these characteristics into three thematic points (use, logical and physical characteristics).

To help you with input location, the characteristics to be entered in the Characteristics of Segment dialog box are marked with **[DB]**.

To implement the access to external resources, you declare the Segments required for the execution of the service and you indicate for each Segment:

- **The logical characteristics**

They make the connection with the Segment description in the VisualAge Pacbase Dictionary.

- the code of the Segment in the VisualAge Pacbase Dictionary **[DB]**
- the type of description **[DB]**:

  blank by default, for a record with a possible common part (e.g. a Data Structure with only one Segment),

  1 for a record without a common part (e.g. for databases, several Segments being defined in the same Data Structure),

  2 for a Segment describing a remote database (ORACLE, SYBASE).

These characteristics are the same *for any type of requested service*: they are therefore declared only once for each Segment.

- The physical characteristics

They identify the physical data storage mode.

- the organization **[DB]**:

  ☞ For the **W** Organization:

  Access generated via specific coding; only a description is generated in the Working-Storage Section.

- the external name **[DB]**:

  DDNAME of the indexed file, or the VisualAge Pacbase code of the Database block in which the table is described, in case of SQL organization.

- the file physical key in **KEY DATA ELEMENT** **[DB]**:

  This information comes under the physical characteristics only for certain organizations, among which the indexed files.

In this case you enter the code of the Data Element which must be specified to access a Segment (it may be a group Data Element).

These characteristics are the same *for any type of requested service*: they are therefore declared only once for each Segment.

- The **use characteristics** specific to a check / update service:

  They represent the nature of the data access and specify an access hierarchy.

  You indicate the type of Segment access required to perform the service as well as the sequence relative to the other Segments' accesses.

  Then the generator produces all the access functions via which the Business Component can perform the selection service for the Logical View.

The use characteristics are made up of the following information:

- the **access sequence** or **access hierarchy**. Access to a Segment is implemented for a given service on a Logical View. You must associate the Segment with the Logical View on which it depends.

  It may be necessary to access several Segments, you can:
  - ♦ associate each Segment with the Logical View:
    In the generated procedures, all check accesses are performed, whatever the results are,
  - ♦ or link the Segments to one another, thus making an access hierarchy:
    In the generated procedures, if a result of a check access is incorrect, the following accesses are not performed.
- the **category** by entering the Segment in one of the three following panels:

  INITIALIZATION

  BODY (iterative)

  For a multi-occurrence Logical View, this category performs an iterative access to the records.

  ☞ This category is **required** for a graphic application.

  TERMINATION
- the **access key source [DB]** contains the origin of the data which allows to specify the access key.
- the **access key [DB]** is the record key or the selection criterion.
- The use type [BD]:

  C consistency check and update by creation only,
  M consistency check and update by modification only,
  S consistency check and update by deletion only,
  T consistency check and update by creation, modification or deletion,
  X consistency check and update by creation or modification.

How to Transfer Data

To implement a check/update/selection service, after the access to the external resources, the data transfer is performed in two directions:

- from the Logical View to the Segments for the check/update service,

- then from the Segments to the Logical View for the selection service.

For details, refer to section Data Transfer.

How to Update Data

The update is automatically carried out if the Segment is assigned with one of the use types previously mentioned and if the Business Component is called for an update service by the Client component.

The update service sends to the storage area (database, file, etc) the data that has been updated in the Client component.

Implementing Selection in the Client Component

The selection allows to display in the Client component the information according to a specified format (number of returned occurrences, number of occurrences displayed in a page...).

If no selection service is specified (i.e. no Segment is specified in the **SEND** panel), only the check/update service will be executed. The information sent to the Client component will be accessed by direct read only.

About how to specify a selection service, refer to *Subchapter* Selection Service.

Structure of the Generated Code

Working Storage Section

CATM	Transaction code:
C	Creation
M	Modification
A	Deletion
X	Implicit update
IRR	Number of repetitions requested by the Client component
ICATRC	Number of requested repetitions during a call of the Business Component
OPERB	Indicates if the selection is a list during a call of the Business Component
OPERT	Area for the management of check/update/selection services

A-CATM	Description buffer. This buffer contains the transaction codes related to both non repeated and repeated data of the Logical View.
A-CATM-CA	Action code applying to non repeated data. Systematically generated.

A-CATM-CR OCCURS N

Action code applying to repeated data.

N is the maximum number of repetitions for all the Logical Views in this server. Generated if one of the Business Component's Logical View contains repeated data.

CH-view **view** represents the Logical View.

This area contains a table used for the checks of the Logical View.

It is generated if the **CHECKSER=YES** option has been specified in the Generation Parameters screen of the Dialogue (or Business Component).

This table contains one item for each Data Element called in the Logical View. The possible values (to be specified by the user in the Client component) are:

N	No check on this Data Element (default value)
P	Missing Data Element
blank	All Data Elements are checked

This area is also used to indicate if the value of a field in the Logical View is null or not (in SQL standard), i.e. if it is present. For example, the Client can set null values for an update service and the Business Component can return null values for the selection (if the corresponding fields in the table are null).

Linkage Section

TECH-ICATRC Number of instances to be processed by the Business Component on the Logical View.

TECH-ICATRS Number of instances selected by the Business Component for a selection or check/update/selection service.

TECH-IDATAC Check indicator of the Logical View data, set by the Client component:

blank Check on the fields of the Logical View whose check Iindicator contains a check request (see the description of the indicator in **CONT-BUFFER**).

N No check

C Check on all the Logical View's fields

SELT-BUFFER End-of-access buffer on a Segment in selection or repetitive category. This buffer contains the following fields:

SELT-SEGCOD Segment code

SELT-SEGERR Code of error on Segment

SELT-SEGTYP Type of error (**S**: standard) on Segment

SELT-LIBRA Library code of the Business Component which has detected the end-of-access on Segment

SELT-SERVER Code of the Business Component which has detected the end-of-access on Segment.

ERR-BUFFER Error buffer, divided into errors on Segments and errors on Data Elements:

ERR-BUFSEG Segment access error buffer:

ERR-SEGCOD Code of the erroneous Segment

ERR-SEGERR Error code:

DUPL Creation by mistake of a record, already existing record

NFND	Modification or deletion by mistake, non-existing record
END	End of list
ABSC	Record to be selected not found
ERR-SEGTYP	Error type
ERR-ICATR	Line number of the erroneous occurrence for a multi-occurrence processing
ERR-LIBRA	Library code
ERR-SERVER	Business Component code
ERR-DATA	Buffer of errors on Data Element
ERR-LIBRA	Library code
ERR-SERVER	Business Component code
ERR-VIEW	Logical View code
ERR-DATCOD	Erroneous Data Element code
ERR-DATERR	Error code
ERR-DATTYP	Error type
ERR-ICATR	Line number of the erroneous occurrence for a multi-occurrence processing.
FILLER	End filler

Procedure Division

For the **'Logical View' block**, tags are as follows:

- the Logical View code in four characters:
  FVIEW
- the type of general processing, in four characters:
  CHUP check and/or update
- the category, in four characters:
  CATR multi-occurrence Logical View
- the service type, in four characters:
  SRVT check / update and selection
  (All the services indicated in the table in section Logical View Processing are generated but only this one will be actually executed if the Client requests a check / update and selection service).
- the type of elementary procedure, in four characters:
  INIT initialization
  CHCK check
  TRAN transfer
  UPDT update
  DONE end

```
20  chck/updt service          FVIEW-CHUP-CATR-SRVM
 25 Initializations            FVIEW-CHUP-CATR-SRVM-INIT
 25 Checks                     FVIEW-CHUP-CATR-SRVM-CHCK
 25 Transfers                  FVIEW-CHUP-CATR-SRVM-TRAN
 25 Update                     FVIEW-CHUP-CATR-SRVM-UPDT
 25 End of service             FVIEW-CHUP-CATR-SRVM-DONE
```

If a Segment has been specified in the **SEND** panel, a branching to the selection service (with **SELC** as type of general processing and **SRVA** as service type) is performed at the end of the check/update service:

```
10     Selection                  FVIEW-SELC
 15    Multi-occurrence process   FVIEW-SELC-CATR
  20   Selection service          FVIEW-SELC-CATR-SRVA
   25 Initializations             FVIEW-SELC-CATR-SRVA-INIT
   25 Selection                   FVIEW-SELC-CATR-SRVA-SELC
   25 Transfers                   FVIEW-SELC-CATR-SRVA-TRAN
   25 End of service              FVIEW-SELC-CATR-SRVA-DONE
```

If no Segment has been specified in the **SEND** panel, only a direct read will be carried out at the end of the check/update service.

For **data transfers**, the check/update/selection service carries out the transfer from the Logical View to the Segment and from the Segment to the Logical View. The tags are as follows:

- the Logical View code in four characters:
  FVIEW
- the transfer direction, in four characters:
  TRDT TRansfer to the "DaTa" in the direction View → Segment
  CHKD Check on the Logical View's Data Elements if option **CHECKSER=YES**.
  Then, for the selection service (if implemented):
  TRVW TRansfer to the "VieW" in the direction Segment → View
- the category, in four characters:
  CATR multi-occurrence Logical View

```
        10     Data transfers to Segment        FVIEW-TRDT
         15    Multi-occurrence transfers       FVIEW-TRDT-CATR
if CHECKSER=YES :
        10     Checks on View data              FVIEW-CHKD
         15    Multi-occurrence checks          FVIEW-CHKD-CATR
          20   Check on Element DELCO1          FVIEW-CHKD-CATR-DELCO1
          20   Check on Element DELCO2          FVIEW-CHKD-CATR-DELCO2
        10     Data transfers to View           FVIEW-TRVW
         15    Multi-occurrence transfers       FVIEW-TRVW-CATR
```

For **logical accesses** (=accesses to Segments required for the execution of the services on the Logical Views), tags are as follows:

- the Segment code in four characters:
  FSEGT
- the type of general processing, in four characters:
  CHCK check
  UPDT update
  SLCT selection
- the category, in four characters:
  CATR multi-occurrence Logical View with a number of occurrences higher than 1
  CATT multi-occurrence Logical View with a number of occurrences to be processed equal to 1 (for selection only)

- the type of elementary processing, in four characters:

ALIM	key loading
CALL	call of the physical accesses (by PERFORM)
ERRS	error handling

```
10     Logical access to Segment for check     FSEGT-CHCK
 15    Multi-occurrence access                 FSEGT-CHCK-CATR
  20   Key loading                             FSEGT-CHCK-CATR-ALIM
  20   Call of physical access                 FSEGT-CHCK-CATR-CALL
  20   Error handling                          FSEGT-CHCK-CATR-ERRS
10     Logical access to Segment for updt      FSEGT-UPDT
 15    Multi-occurrence access                 FSEGT-UPDT-CATR
  20   Key loading                             FSEGT-UPDT-CATR-ALIM
  20   Call of physical access                 FSEGT-UPDT-CATR-CALL
  20   Error handling                          FSEGT-UPDT-CATR-ERRS
10     Logical access to Segment for slct      FSEGT-SLCT
 15    Multi-occurrence access                 FSEGT-SLCT-CATR
  20   Key loading                             FSEGT-SLCT-CATR-ALIM
  20   Call of physical access                 FSEGT-SLCT-CATR-CALL
  20   Error handling                          FSEGT-SLCT-CATR-ERRS
 15    Access to one occurrence                FSEGT-SLCT-CATT
  20   Key loading                             FSEGT-SLCT-CATT-ALIM
  20   Call of physical access                 FSEGT-SLCT-CATT-CALL
  20   Error handling                          FSEGT-SLCT-CATT-ERRS
```

For **physical accesses**, tags are as follows:

- F80
- the Segment code in four characters: SEGT
- the access code:

R	Read for check
RA	Read for selection (if selection implemented)
RU	Read for check before update
P	Read of first record (if selection implemented)
RN	Read of next record (if selection implemented)
W	Creation (write)
RW	Modification (rewrite)
D	Deletion (Delete)
UN	Unlock of record read in RU

```
10     Read for check                          F80-SEGT-R
10     Read for selection                      F80-SEGT-RA
10     Read for check before update            F80-SEGT-RU
10     Read of first record                    F80-SEGT-P
10     Read of next record                     F80-SEGT-RN
10     Creation (write)                        F80-SEGT-W
10     Modification (rewrite)                  F80-SEGT-RW
10     Deletion (delete)                       F80-SEGT-D
10     Unlock of record read in RU             F80-SEGT-UN
```

Inserting Specific Code

VisualAge Pacbase allows you to implement processes that are not standard processes. For examples of inserting specific code, refer to the corresponding paragraphs describing the check, update and selection services.

For the general rules for inserting specific code, refer to sections General Principles and Important Rules.

You first create a function where you identify the Logical View by its code. This approach is common to all the specific processes in the Logical View. This is why it is described in section Insertion Relative to the Logical View Processing.

Then, for each specific procedure to be inserted, you specify the procedure relative to a standard procedure of the Logical View. To do so, create a sub-function. Its code, title, hierarchical level and type of setting are common to all the specific procedures of the Logical View. This is why they are described in section Insertion Relative to the Logical View Processing.

Specific code can be inserted:

- In the Logical View processing (**FVIEW-CHUP** and **FVIEW-SELC** tags).
  - relative to the level 10 for check/update/selection serviceIf you insert specific code at this level, it will work for all the Check/Update services of the Logical View. This is the most generic specific code.

    The input is as follows:

    ```
    OPE OPERAND                          TY CONDITION
    Function XX  Sub-function   YY   Level 10
     N SS-FCT TITLE                         type genp
    ```

    type represents the insertion type: ***R** to replace the standard processing, ***A** or ***P** to add a specific processing to the standard one. The level 10 is not mandatory for ***A** and ***P** types. **genp** represents the general processing (**CHUP** for check/update or **SELC** for selection).
  - relative to the level **15** for check/update/selection processing by category.

    The input is as follows:

    ```
    OPE OPERAND                          TY CONDITION
    Function XX  Sub-function YY       Level 15
    N   SUB-FCT TITLE                      type genp CATR
    ```

    Same comments as for the insertion at level **10**. The **CATR** category is the only one possible for a check/update/selection service.
  - relative to the level **20** for check/update/selection processing by category and service type.

    The input is as follows:

    ```
    OPE OPERAND                          TY CONDITION
    Function XX  Sub-function YY       Level 20
    N   SUB-FCT TITLE                  type CHUP CATR SRVT
    or
    N   SUB-FCT TITLE                  type SELC CATR SRVA
    ```

    Same comments as for the previous insertion. **SRVT** represents the check / update / selection service. **SRVA** represents the selection service.
- In the data transfers from Logical View to Segment and from Segment to Logical View (**FVIEW-TRDT** and **FVIEW-TRVW** tag):
  - Relative to level **10** for the transfer direction:

    The input is as follows:

    ```
    OPE OPERAND                          TY CONDITION
    Function XX  Sub-function YY   Level 10
     N   SUB-FCT TITLE                      type DRCT
    ```

type represents the insertion type: *R to replace the standard processing, *A or *P to add a specific processing to the standard one. The level 10 is not mandatory for *A and *P types. DRCT represents the transfer direction: TRDT for the check/update service and TRVW for the selection service.

- relative to the level 15 for the check/update/selection service by category.

  The input is as follows:

```
OPE OPERAND                          TY CONDITION
Function XX   Sub-function YY     Level 15
N SUB-FCT TITLE                          type drct catg
```

Same comments as for the insertion at level 10.

catg represents the category :

CATR for a multi-occurrence Logical View if the number of occurrences to be processed is higher than 1,

CATT if the number of occurrence is equal to 1 (for the selection service only).

- In the logical accesses (FSEGT-CHCK, FSEGT-UPDT and FSEGT-SLCT tags):

  - relative to the level 10 for the access to a Segment for data check/update/selection:

    The input is as follows:

```
OPE OPERAND                          TY CONDITION
Function XX   Sub-function YY     Level 10
 N SUB-FCT TITLE                         type genp segt
```

type represents the insertion type:

*R to replace the standard processing,

*A or *P to add a specific processing to the standard one. The level 10 is not mandatory for *A and *P types.

genp represents the type of general processing: CHCK, UPDT or SLCT.

segt represents the Segment code.

  - relative to the level 15 for the access to a Segment for data check/update/selection by category.

    The input is as follows:

```
OPE OPERAND                          TY CONDITION
Function XX   Sub-function YY     Level 15
 N SUB-FCT TITLE                      type genp segt catg
```

Same comments as for the insertion at level 10.

catg represents the category:

CATR is required for check and update,

CATT also possible for selection.

  - relative to the level 20 for the access to a Segment for data check/update/selection by category and elementary service type.

    The input is as follows:

```
OPE OPERAND                        TY CONDITION
Function XX   Sub-function YY      Level 20
 N SUB-FCT TITLE                 type genp segt catg elmp
```

Same comments as for the previous insertion.

elmp represents the type of elementary processing (ALIM for key loading, CALL for the call of physical processing, ERRS error processing).

- In the physical accesses (**F80-SEGT-R, F80-SEGT-RA, F80-SEGT-RU, F80-SEGT-P, F80-SEGT-RN, F80-SEGT-W, F80-SEGT-RW, F80-SEGT-D, F80-SEGT-UN** tags).

  👓 Refer to section Insertion Relative to the 'Physical Accesses' Level.

Check and Selection

A check and selection service is needed when you want to check the data entered by the user (if necessary, with the display of an error message) and present in the Client component the information according to a specified format, without updating the database.

Implementation

To implement a check and selection service, you must:

- declare the Logical View containing the service request issued by the Client component,
- perform the access to the external resources to check data consistency.
- carry out the selection in the Client component.

Reminder In the Dictionary, the description of external resources is specified on Segment type occurrences.

How to Declare the Logical View

The Business Component must know the **Logical View** for which it implements services.

You declare the Logical View in the Business Component's **Call of Segments (semi local)** window.

👓 The use of this window (menus, dialog boxes, etc.) is detailed in a specific chapter of the *VisualAge Pacbase WorkStation* User Interface Guide. In addition, complete documentation for all input fields is found in the *Pacbench C/S – Business Logic & TUI Clients Manual*.

How to Access External Resources

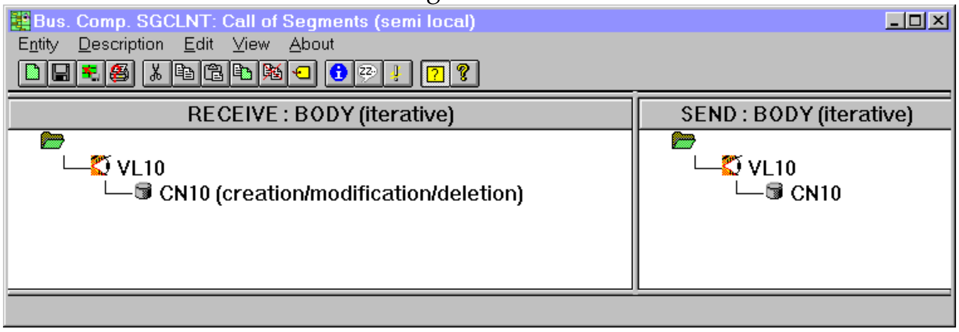
A check and selection service may require access to several external resources.

To implement a check and selection service, you specify the Segment code in the **Call of Segments (semi local)** window:

- in the **RECEIVE** panel for the Business Component receives from the Client component, the data to be checked and updated via the Logical View,
- and in the **SEND** panel for the Business Component sends the data to the Client component to refresh its information.

If no Segment is entered in the **SEND** panel, the selection service will not be executed and only a direct read will be performed.

The Segment(s) used as reference(s) for check can be different from that(those) used as reference(s) for selection but all of them must be associated with the same Logical View.


Bus. Comp. SGCLNT: Call of Segments (semi local)

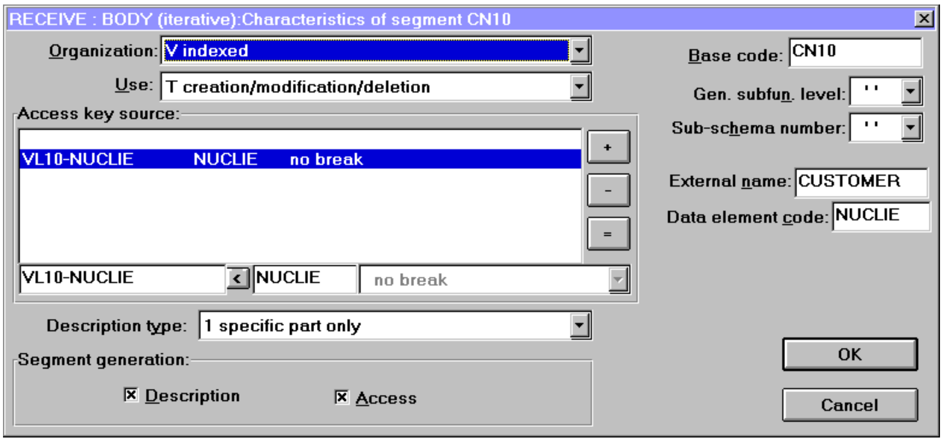
RECEIVE : BODY (iterative)
VL10
CN10 (creation/modification/deletion)

SEND : BODY (iterative)
VL10
CN10

The following comments apply to the access to the external resources for the check/update service (i.e to the Segment indicated in the **RECEIVE** panel). For the specification of the access to the external resources for the selection service, refer to sectionHow to Access External Resources

.

You specify the characteristics of the Segment in the **Characteristics of Segment** dialog box, that you open by double-clicking on the Segment code.


RECEIVE : BODY (iterative):Characteristics of segment CN10

Organization: V indexed
Use: T creation/modification/deletion
Access key source:
VL10-NUCLIE NUCLIE no break
VL10-NUCLIE NUCLIE no break
Description type: 1 specific part only
Segment generation:
☒ Description ☒ Access

Base code: CN10
Gen. subfun. level: ' '
Sub-schema number: ' '
External name: CUSTOMER
Data element code: NUCLIE

For a better functional understanding, we have grouped these characteristics into three thematic points (use, logical and physical characteristics).

To help you with input location, the characteristics to be entered in the Characteristics of Segment dialog box are marked with **[DB]**.

To implement the access to external resources, you declare the Segments required for the execution of the service and you indicate for each Segment:

- **The logical characteristics**

They make the connection with the Segment description in the VisualAge Pacbase Dictionary.

- the code of the Segment in the VisualAge Pacbase Dictionary **[DB]**
- the type of description **[DB]**:

blank by default, for a record with a possible common part (e.g. a Data Structure with only one Segment),

1 for a record without a common part (e.g. for databases, several Segments being defined in the same Data Structure),
2 for a Segment describing a remote database (ORACLE, SYBASE).

These characteristics are the same *for any type of requested service*: they are therefore declared only once for each Segment.

- The physical characteristics

They identify the physical data storage mode.

- the organization **[DB]**:
  - ☞ For the **W** Organization:

    Access generated via specific coding; only a description is generated in the Working-Storage Section.
- the external name **[DB]**:

  DDNAME of the indexed file, or the VisualAge Pacbase code of the Database block in which the table is described, in case of SQL organization.
- the file physical key in **KEY DATA ELEMENT** **[DB]**:

  This information comes under the physical characteristics only for certain organizations, among which the indexed files.

  In this case you enter the code of the Data Element which must be specified to access a Segment (it may be a group Data Element).

  These characteristics are the same *for any type of requested service*: they are therefore declared only once for each Segment.

- The **use characteristics** specific to a check / update service:

  They represent the nature of the data access and specify an access hierarchy.

  You indicate the type of Segment access required to perform the service as well as the sequence relative to the other Segments' accesses.

  Then the generator produces all the access functions via which the Business Component can perform the selection service for the Logical View.

The use characteristics are made up of the following information:

- the **access sequence** or **access hierarchy**. Access to a Segment is implemented for a given service on a Logical View. You must associate the Segment with the Logical View on which it depends.

  It may be necessary to access several Segments, you can:
  - ♦ associate each Segment with the Logical View:
    In the generated procedures, all check accesses are performed, whatever the results are,
  - ♦ or link the Segments to one another, thus making an access hierarchy:
    In the generated procedures, if a result of a check access is incorrect, the following accesses are not performed.
- the **category** by entering the Segment in one of the three following panels:

  INITIALIZATION

BODY (iterative)

For a multi-occurrence Logical View, this category performs an iterative access to the records.

☞ This category is **required** for a graphic application.

TERMINATION

- the **access key source [DB]** contains the origin of the data which allows to specify the access key.
- the **access key [DB]** is the record key or the selection criterion.

- the use type [BD]:

C consistency check and update by creation only,
E existence check only,
M consistency check and update by modification only,
S consistency check and update by deletion only,
T consistency check and update by creation, modification or deletion,
X consistency check and update by creation or modification.

How to Implement Selection in the Client component

The selection allows to display in the Client component the information according to a specified format (number of returned occurrences, number of occurrences displayed in a page...).

If no selection service is specified (i.e. no Segment is specified in the **SEND** panel), only the check/update service will be executed. The information sent to the Client component will be accessed by direct read only.

About how to specify a selection service, refer to *Subchapter* Selection Service.

Structure of the Generated Code

Working Storage Section

IRR Number of repetitions requested by the Client component

ICATRC Number of requested repetitions during a call of the Business Component

OPERB Indicates if the selection is a list during a call of the Business Component

OPERT Area for the management of check/update/selection services

CH-view **view** represents the Logical View.

This area contains a table used for the checks of the Logical View.

It is generated if the **CHECKSER=YES** option has been specified in the Generation Parameters screen of the Dialogue (or Business Component).

This table contains one item for each Data Element called in the Logical View. The possible values (to be specified by the user in the Client component) are:

N No check on this Data Element (default value)

`P` Missing Data Element

`blank` All Data Elements are checked

This area is also used to indicate if the value of a field in the Logical View is null or not (in SQL standard), i.e. if it is present. For example, the Client can set null values for an update service and the Business Component can return null values for the selection (if the corresponding fields in the table are null).

Linkage Section

`TECH-ICATRC` Number of instances to be processed by the Business Component on the Logical View.

`TECH-ICATRS` Number of instances selected by the Business Component for a selection or check/update/selection service.

`TECH-IDATAC` Check indicator of the Logical View data, set by the Client component:

`blank` Check on the fields of the Logical View whose check indicator contains a check request (see the description of the indicator in `CONT-BUFFER`).

`N` No check

`C` Check on all the Logical View's fields

`SELT-BUFFER` End-of-access buffer on a Segment in selection or repetitive category. This buffer contains the following fields:

`SELT-SEGCOD` Segment code

`SELT-SEGERR` Code of error on Segment

`SELT-SEGTYP` Type of error (`S`: standard) on Segment

`SELT-LIBRA` Library code of the Business Component which has detected the end-of-access on Segment

`SELT-SERVER` Code of the Business Component which has detected the end-of-access on Segment.

`ERR-BUFFER` Error buffer, divided into errors on Segments and errors on Data Elements:

`ERR-BUFSEG` Segment access error buffer:

`ERR-SEGCOD` Code of the erroneous Segment

`ERR-SEGERR` Error code:

`DUPL` Creation by mistake of a record, already existing record

`NFND` Modification or deletion by mistake, non-existing record

`END` End of list

`ABSC` Record to be selected not found

`ERR-SEGTYP` Error type

`ERR-ICATR` Line number of the erroneous occurrence for a multi-occurrence processing

`ERR-LIBRA` Library code

ERR-SERVER	Business Component code
ERR-DATA	Buffer of errors on Data Element
ERR-LIBRA	Library code
ERR-SERVER	Business Component code
ERR-VIEW	Logical View code
ERR-DATCOD	Erroneous Data Element code
ERR-DATERR	Error code
ERR-DATTYP	Error type
ERR-ICATR	Line number of the erroneous occurrence for a multi-occurrence processing.
FILLER	End filler

Procedure Division

For the **'Logical View' block**, tags are as follows:

- the Logical View code in four characters:
  FVIEW
- the type of general processing, in four characters:
  CHUP check and/or update
- the category, in four characters:
  CATR multi-occurrence Logical View
- the service type, in four characters:
  SRVX check and selection
  (All the services indicated in the table in section Logical View Processing are generated but only this one will be actually executed if the Client requests a check and selection service).
- the type of elementary procedure, in four characters:
  INIT initialization
  CHCK check
  DONE end

```
 20  Check/selection service    FVIEW-CHUP-CATR-SRVX
  25 Initializations            FVIEW-CHUP-CATR-SRVX-INIT
  25 Checks                     FVIEW-CHUP-CATR-SRVX-CHCK
  25 End of service             FVIEW-CHUP-CATR-SRVX-DONE
```

If a Segment has been specified in the SEND panel, a branching to the selection service (with SELC as type of general processing and SRVA as service type) is performed at the end of the check/update service:

```
10     Selection                FVIEW-SELC
 15    Multi-occurrence process FVIEW-SELC-CATR
  20   Selection service        FVIEW-SELC-CATR-SRVA
   25 Initializations           FVIEW-SELC-CATR-SRVA-INIT
   25 Selection                 FVIEW-SELC-CATR-SRVA-SELC
   25 Transfers                 FVIEW-SELC-CATR-SRVA-TRAN
   25 End of service            FVIEW-SELC-CATR-SRVA-DONE
```

If no Segment has been specified in the **SEND** panel, only a direct read will be carried out at the end of the check/update service.

The check and update service performs the **data transfers** from the Segment to the Logical View. The tags are as follows:

- the Logical View code in four characters: **FVIEW**
- the possible data check, in four characters:

  CHKD Check on the Logical View's Data Elements if the **CHECKSER=YES**.

  Then, transfer direction for the selection service (if implemented):

  TRVW TRansfer to "VieW" in the direction Segment → View
- the category, in four characters:

  CATR multi-occurrence Logical View

if **CHECKSER=YES** :

```
10    Checks on View data            FVIEW-CHKD
 15   Multi-occurrence checks        FVIEW-CHKD-CATR
  20  Check on Element DELCO1        FVIEW-CHKD-CATR-DELCO1
  20  Check on Element DELCO2        FVIEW-CHKD-CATR-DELCO2
10    Data transfers to View         FVIEW-TRVW
 15   Multi-occurrence transfers     FVIEW-TRVW-CATR
```

For **logical accesses** (accesses to Segments required for the execution of the services on the Logical Views), tags are as follows:

- the Segment code in four characters:

  FSEGT
- the type of general processing, in four characters:

  CHCK check

  SLCT selection
- the category, in four characters:

  CATR multi-occurrence Logical View with a number of occurrences higher than 1

  CATT multi-occurrence Logical View with a number of occurrences to be processed equal to 1 (for selection only)
- the type of elementary processing, in four characters:

  ALIM key loading

  CALL call of the physical accesses (by **PERFORM**)

  ERRS error handling

```
10    Logical access to Segment for check    FSEGT-CHCK
 15   Multi-occurrence access                FSEGT-CHCK-CATR
  20  Key loading                            FSEGT-CHCK-CATR-ALIM
  20  Call of physical access                FSEGT-CHCK-CATR-CALL
  20  Error handling                         FSEGT-CHCK-CATR-ERRS
10    Logical access to Segment for slct     FSEGT-SLCT
 15   Multi-occurrence access                FSEGT-SLCT-CATR
  20  Key loading                            FSEGT-SLCT-CATR-ALIM
  20  Call of physical access                FSEGT-SLCT-CATR-CALL
  20  Error handling                         FSEGT-SLCT-CATR-ERRS
 15   One-occurrence access                  FSEGT-SLCT-CATT
  20  Key loading                            FSEGT-SLCT-CATT-ALIM
  20  Call of physical access                FSEGT-SLCT-CATT-CALL
  20  Error handling                         FSEGT-SLCT-CATT-ERRS
```

For **physical accesses**, tags are as follows:

- **F80**
- the Segment code in four characters: **SEGT**

- the access code:

R	Read for check	
RA	Read for selection	(if selection implemented)
P	Read of first record	(if selection implemented)
RN	Read of next record	(if selection implemented)

```
10    Read for check                 F80-SEGT-R
10    Read for selection             F80-SEGT-RA
10    Read of first record           F80-SEGT-P
10    Read of next record            F80-SEGT-RN
```

Inserting Specific Code

VisualAge Pacbase allows you to implement processes that are not standard processes.

For examples of inserting specific code, refer to the corresponding paragraphs describing the check, update and selection services.

For the general rules for inserting specific code, refer to sections General Principles and Important Rules.

You first create a function where you identify the Logical View by its code. This approach is common to all the specific processes in the Logical View. This is why it is described in section Insertion Relative to the Logical View Processing.

Then, for each specific procedure to be inserted, you specify the procedure relative to a standard procedure of the Logical View. To do so, create a sub-function. Its code, title, hierarchical level and type of setting are common to all the specific procedures of the Logical View. This is why they are described in section Insertion Relative to the Logical View Processing.

Specific code can be inserted:

- In the Logical View processing (FVIEW-CHUP and FVIEW-SELC tags).
  - relative to the level 10 for check/ and selection service. If you insert specific code at this level, it will work for all the Check/Update services of the Logical View. This is the most generic specific code.

  The input is as follows:

```
OPE OPERAND                          TY CONDITION
Function XX   Sub-function   YY   Level 10
 N SS-FCT TITLE                         type genp
```

  type represents the insertion type: *R to replace the standard processing, *A or *P to add a specific processing to the standard one. The level 10 is not mandatory for *A and *P types. genp represents the general processing (CHUP for check/update or SELC for selection).

  - relative to the level 15 for check and selection processing by category.

    The input is as follows:

```
OPE OPERAND                          TY CONDITION
Function XX   Sub-function YY       Level 15
N   SUB-FCT TITLE                        type genp CATR
```

    Same comments as for the insertion at level 10. CATR is the only possible category for a check/selection service.

- relative to the level 20 for check/selection processing by category and service type.

  The input is as follows:

  ```
  OPE OPERAND                          TY CONDITION
  Function XX   Sub-function YY    Level 20
  N   SUB-FCT TITLE                  type CHUP CATR SRVX
  ```

  or

  ```
  N   SUB-FCT TITLE                  type SELC CATR SRVA
  ```

  Same comments as for the previous insertion. **SRVX** represents the check service. **SRVA** represents the selection service.

- In the data transfers from the Segment to the Logical View (**FVIEW-TRVW** tag):

  Relative to level **10** for the transfer direction:

  The input is as follows:

  ```
  OPE OPERAND                          TY CONDITION
  Function XX   Sub-function YY   Level 10
   N   SUB-FCT TITLE                     type TRVW
  ```

  type represents the insertion type: ***R** to replace the standard processing, ***A** or ***P** to add a specific processing to the standard one. The level **10** is not mandatory for ***A** and ***P** types.

  - relative to the level **15** for the transfer direction by category.

    The input is as follows:

    ```
    OPE OPERAND                          TY CONDITION
    Function XX   Sub-Function YY   Level 15
     N SUB-FCT TITLE                     type TRVW CATR
    ```

    Same comments as for the insertion at level **10**. The **CATR** category is the only one possible.

- In the logical accesses (**FSEGT-CHCK** and **FSEGT-SLCT** tags):

  - relative to the level **10** for the access to a Segment for data check/selection:

    The input is as follows:

    ```
    OPE OPERAND                          TY CONDITION
    Function XX   Sub-function YY     Level 10
     N SUB-FCT TITLE                     type genp segt
    ```

    type represents the insertion type: ***R** to replace the standard processing, ***A** or ***P** to add a specific processing to the standard one. The level 10 is not mandatory for ***A** and ***P** types. **genp** represents the type of general processing: **CHCK**, **UPDT** or **SLCT**. **segt** represents the Segment code.

  - relative to the level **15** for the access to a Segment for data check/selection by category.

    The input is as follows:

    ```
    OPE OPERAND                          TY CONDITION
    Function XX   Sub-function YY     Level 15
     N SUB-FCT TITLE                  type genp segt catg
    ```

    Same comments as for the insertion at level **10**.

    catg represents the category: **CATR** is required for check and update and **CATT** also possible for selection.

  - relative to the level **20** for the access to a Segment for data check/selection by category and elementary service type.

    The input is as follows:

    ```
    OPE OPERAND                       TY CONDITION
    ```

```
Function XX   Sub-function YY     Level 20
 N SUB-FCT TITLE                 type genp segt catg elmp
```

Same comments as for the previous insertion. `elmp` represents the type of elementary processing (`ALIM` for key loading, `CALL` for the call of physical processing, `ERRS` error processing).

- In the physical accesses (`F80-SEGT-R`, `F80-SEGT-RA`, `F80-SEGT-P`, `F80-SEGT-RN` tags).

Refer to section Insertion Relative to the 'Physical Accesses' Level.

Selection Service

A selection service accesses the data stored in a database or a file. It reads the information requested by the Client component via the Logical View and sends it back. This information is presented in the Client component according to a specified format (number of lines per page…).

The selection service returns the number of requested instances, unless the number is higher than the iterative capacity of the Logical View.

The selection service is a service in display: data are transferred from the Segment (representing a Table, a Record…) to the Logical View.

Implementation

To implement a **selection service**, you must:

- declare the Logical View containing the service request issued by the Client component,
- perform the access to the external resources to select the record(s)
- implement the transfer of information from the record(s) to the Logical View.

Reminder In the Dictionary, the description of external resources is specified on Segment type occurrences.

How to Declare the Logical View

The Business Component must know the **Logical View** for which it implements services.

You declare the Logical View in the Business Component's `Call of Segments (semi local)` window.

The use of this window (menus, dialog boxes, etc.) is detailed in a specific chapter of the *VisualAge Pacbase WorkStation* User Interface Guide. In addition, complete documentation for all input fields is found in the *Pacbench C/S – Business Logic & TUI Clients Manual.*

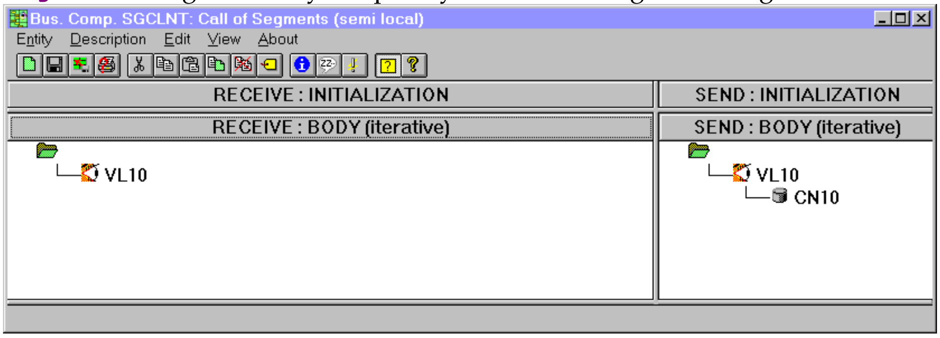
How to Access External Resources

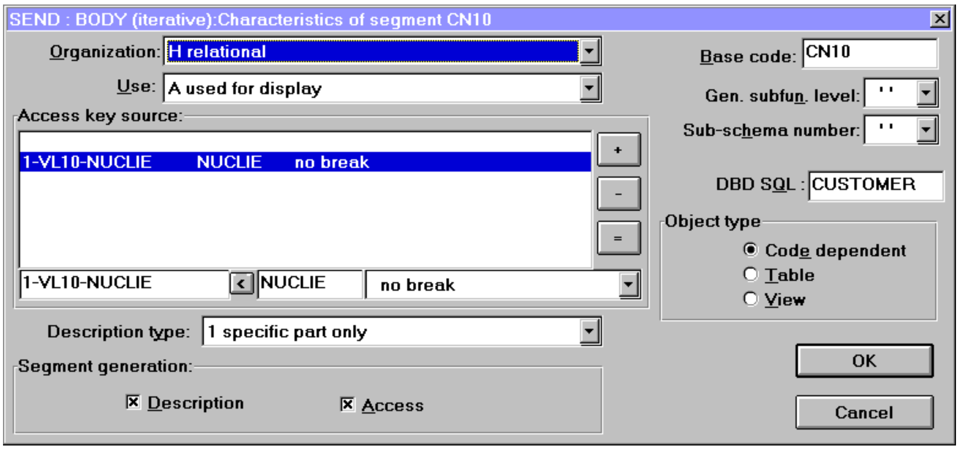
The selection of information requires access to one or more external resources (Segment, table, record, etc.).

Reminder In the Dictionary, the description of external resources is specified on Segment type occurrences.

To implement a selection service, you specify the Segment code in the `Call of Segments (semi local)` window, in the `SEND` panel. Indeed, the Business Component receives from the Client component, the data to be displayed via the Logical View.

You enter the characteristics of the Segment in the `Characteristics of segment` dialog box that you open by double-clicking on the Segment code.


Bus. Comp. SGCLNT: Call of Segments (semi local)
Entity Description Edit View About
RECEIVE : INITIALIZATION
SEND : INITIALIZATION
RECEIVE : BODY (iterative)
SEND : BODY (iterative)
VL10
VL10
CN10



Organization: H relational

Use: A used for display

Access key source:

1-VL10-NUCLIE NUCLIE no break

1-VL10-NUCLIE NUCLIE no break

Description type: 1 specific part only

Segment generation:

☒ Description ☒ Access

Base code: CN10

Gen. subfun. level: ' '

Sub-schema number: ' '

DBD SQL : CUSTOMER

Object type

● Code dependent

○ Table

○ View

For a better functional understanding, we have grouped these characteristics into three thematic points (use, logical and physical characteristics).

To help you with input location, the characteristics to be entered in the `Characteristics of Segment` dialog box are marked with [DB].

To implement the access to external resources, you declare the Segments required for the execution of the service and you indicate for each Segment:

- The **logical** characteristics

They make the connection with the Segment description in the VisualAge Pacbase Dictionary.

  - the code of the Segment in the Business Component

- the code of the Segment in the VisualAge Pacbase Dictionary **[DB]**
- the description type **[DB]**:

  `blank` by default, for a record with a possible common part (e.g. a Data Structure with only one Segment),

  `1` for a record without a common part (e.g. for databases, several Segments being defined in the same Data Structure),

  `2` for a Segment describing a remote database (ORACLE, SYBASE).

These characteristics are the same for any type of requested service: they are therefore declared only once for each Segment.

- The **physical** characteristics

They identify the physical data storage mode.

- the organization **[DB]**:

  ☞ For the `W` Organization:

  Access generated via specific coding; only a description is generated in the Working-Storage Section.

  This is also the value to select for the Segment describing a User Buffer. This selection is made in the Business Components' *Dialogue*.

- the external name **[DB]**:

  `DDNAME` of the indexed file, or the VisualAge Pacbase code of the Database block in which the table is described, in case of SQL organization.

- the file physical key in `KEY DATA ELEMENT` **[DB]**:

  This information comes under the physical characteristics only for certain organizations, among which the indexed files.

  In this case you enter the code of the Data Element which must be specified to access a Segment (it may be a group Data Element).

These characteristics are the same for any type of requested service: they are therefore declared only once for each Segment.

- The **use** characteristics:

They represent the nature of the data access and specify an access hierarchy.

You indicate the type of Segment access required to perform the service as well as the sequence relative to the other Segments' accesses, if needed.

Then the generator produces all the access functions via which the Business Component can perform the selection service for the Logical View.

The use characteristics are made up of the following information:

- the **access sequence** or **access hierarchy**. Access to a Segment is implemented for a given service on a Logical View. You must associate the Segment with the Logical View on which it depends.

  It may be necessary to access several Segments. You must then associate the first Segment to the Logical View and link the other Segments to the first one. The access key source of the other Segments is necessarily a field of the first Segment.

In other words, you build an access hierarchy, called "navigational tree".

In the generated code, the records of the Segment associated with the Logical View are read in a sequential-reading iterative loop. At each iteration, the records of the other Segments are read by direct access from the record of the first Segment.

- the **category** by entering the Segment in one of the three following panels:
  INITIALIZATION
  BODY (iterative)
  For a multi-occurrence Logical View, this category performs an iterative access to the records.
  ☞ This category is **required** for a graphic application.
  TERMINATION
- the **access key source** **[DB]** contains the origin of the data which allows to specify the access key.
- the **access key** **[DB]** is the record key or the selection criterion.
- The **use type** **[BD]** for a selection service:

A	**USE FOR DISPLAY** Read with information selection.

- **selection break** **[DB]**: the key Data Element remains constant during the selection.

blank	no break
E	SQL-specific selection break
C	Selection break
R	Large reading

  This option is available with Segments with an **H**, **D**, or **V** organization.
  To specify a large reading (e.g. the display of all the lines of all the orders), you open the **Characteristics of Segment** dialog box for *each* Segment associated with the Logical View.
  You must enter the value **R** for each key Data Element corresponding to a key Data Element of the Logical View (*belonging to the Folder root node if a Folder has been implemented*).

How to Transfer Data

To implement a check/update/selection service, after the access to the external resources, the data transfer is performed from the Segments to the Logical View's data:

For details, refer to section Data Transfer.

Structure of the Generated Code

Working Storage Section

IRR Number of repetitions requested by the Client component

ICATRC Number of requested repetitions during a call of the Business Component

OPERB Indicates if the selection is a list during a call of the Business Component

OPERT Area for the management of check/update/selection services

CH-view **view** is the Logical View.
This area is also used to indicate if the value of a field in the Logical View is null or not (in SQL standard), i.e. if it is present. For example, the Client can set null values for an update service and the Business Component can return null values for the selection (if the corresponding fields in the table are null).

Linkage Section

TECH-ICATRS Number of instances selected by the Business Component for a selection or check/update/selection service.

SELT-BUFFER End-of-access buffer on a Segment in selection or repetitive category. This buffer contains the following fields:

SELT-SEGCOD Segment code

SELT-SEGERR Code of error on Segment

SELT-SEGTYP Type of error (**S**: standard) on Segment

SELT-LIBRA Library code of the Business Component which has detected the end-of-access on Segment

SELT-SERVER Code of the Business Component which has detected the end-of-access on Segment.

Procedure Division

For the **'Logical View' block**, tags generated between **FVIEW-BEGV** and **FVIEW-ENDV** are as follows:

- the Logical View code in four characters:
  FVIEW
- the type of general processing, in four characters:
  SELC selection
- the category, in four characters:
  CATR multi-occurrence Logical View
- the service type, in four characters:
  SRVA selection
- the type of elementary procedure, in four characters:
  INIT initialization
  SELC selection
  TRAN data transfer
  DONE end

```
10     Selection                  FVIEW-SELC
 15    Multi-occurrence process   FVIEW-SELC-CATR
  20   Selection service          FVIEW-SELC-CATR-SRVA
   25 Initializations             FVIEW-SELC-CATR-SRVA-INIT
   25 Selection                   FVIEW-SELC-CATR-SRVA-SELC
   25 Transfers                   FVIEW-SELC-CATR-SRVA-TRAN
   25 End of service              FVIEW-SELC-CATR-SRVA-DONE
```

The selection service carries out the **data transfers** from the Segment to the Logical View. The tags are as follows:

- the Logical View code in four characters: `FVIEW`
- the transfer direction, in four characters:
  `TRVW` TRansfer to "VieW" in the direction Segment → View
- the category, in four characters:
  `CATR` multi-occurrence Logical View

```
10     Data transfers to View                  FVIEW-TRVW
 15    Multi-occurrence transfers              FVIEW-TRVW-CATR
```

For **logical accesses** (=accesses to Segments required for the execution of the services on the Logical Views), tags are as follows:

- the Segment code in four characters:
  `FSEGT`
- the type of general processing, in four characters:
  `SLCT` selection

- the category, in four characters:
  `CATR` multi-occurrence Logical View with a number of occurrences to be processed higher than 1
  `CATT` multi-occurrence Logical View with a number of occurrences to be processed equal to 1

- the type of elementary processing, in four characters:
  `ALIM` key loading
  `CALL` call of the physical accesses (by `PERFORM`)
  `ERRS` error handling

```
10     Logical access to Segment for slct      FSEGT-SLCT
 15    Multi-occurrence access                 FSEGT-SLCT-CATR
  20   Key loading                             FSEGT-SLCT-CATR-ALIM
  20   Call of physical access                 FSEGT-SLCT-CATR-CALL
  20   Error handling                          FSEGT-SLCT-CATR-ERRS
 15    One-occurrence access                   FSEGT-SLCT-CATT
  20   Key loading                             FSEGT-SLCT-CATT-ALIM
  20   Call of physical access                 FSEGT-SLCT-CATT-CALL
  20   Error handling                          FSEGT-SLCT-CATT-ERRS
```

For **physical accesses**, tags are as follows:

- `F80`
- the Segment code in four characters:
  `SEGT`
- the access code:
  `RA` Read for selection
  `P` Read of first record
  `RN` Read of next record

```
10                Read for selection          F80-SEGT-RA
10                Read of first record        F80-SEGT-P
10                Read of next record         F80-SEGT-RN
```

Inserting Specific Code

VisualAge Pacbase allows you to implement processes that are not standard processes.

In a selection service, you can choose to display some order lines by taking into account the orders of one given month only. You can also read all the order lines for a customer and add up all the amounts.

For the general rules for inserting specific code, refer to sections General Principles and Important Rules.

You first create a function where you identify the Logical View by its code. This approach is common to all the specific processes in the Logical View. This is why it is described in section Insertion Relative to the Logical View Processing.

Then, for each specific procedure to be inserted, you specify the procedure relative to a standard procedure of the Logical View. To do so, create a sub-function. Its code, title, hierarchical level and type of setting are common to all the specific procedures of the Logical View. This is why they are described in section Insertion Relative to the Logical View Processing.

Specific code can be inserted:

- In the Logical View processing (**FVIEW-SELC** tag).
  - relative to the level **10** for the selection service. If you insert specific code at this level, it will work for all the Check/Update services of the Logical View. This is the most generic specific code.

    ```
    OPE OPERAND                          TY CONDITION
    Function XX   Sub-function   YY   Level 10
     N SS-FCT TITLE                        type SELC
    ```

    type represents the insertion type: ***R** to replace the standard processing, ***A** or ***P** to add a specific processing to the standard one. The level 10 is not mandatory for ***A** and ***P** types.
  - relative to the level **15** for selection processing by category.

    The input is as follows:

    ```
    OPE OPERAND                          TY CONDITION
    Function XX   Sub-function YY      Level 15
    N   SUB-FCT TITLE                    type SELC CATR
    ```

    Same comments as for the insertion at level **10**.

    CATR is the only possible category for a selection service.
  - relative to the level **20** for selection processing by category and service type.

    The input is as follows:

    ```
    OPE OPERAND                          TY CONDITION
    Function XX   Sub-function YY    Level 20
     N   SUB-FCT TITLE                 type SELC CATR SRVA
    ```

    Same comments as for the previous insertion.

    SRVA represents the selection service.

- In the data transfers from the Segment to the Logical View (**FVIEW-TRVW** tag):

  Relative to level **10** for the transfer direction:

    The input is as follows:

    ```
    OPE OPERAND                          TY CONDITION
    Function XX   Sub-function YY   Level 10
     N   SUB-FCT TITLE                     type TRVW
    ```

    type represents the insertion type: ***R** to replace the standard processing, ***A** or ***P** to add a specific processing to the standard one. The level **10** is not mandatory for ***A** and ***P** types.

  - relative to the level **15** for the transfer direction by category.

    The input is as follows:

    ```
    OPE OPERAND                          TY CONDITION
    Function XX   Sub-Function YY   Level 15
     N SUB-FCT TITLE                      type TRVW CATR
    ```

    Same comments as for the insertion at level **10**. The **CATR** category is the only one possible.

- In the logical accesses (**FSEGT-SLCT** tag):
  - relative to the level **10** for the access to a Segment for data selection:

    The input is as follows:

    ```
    OPE OPERAND                          TY CONDITION
    Function XX   Sub-function YY     Level 10
     N SUB-FCT TITLE                      type SLCT segt
    ```

    type represents the insertion type: ***R** to replace the standard processing, ***A** or ***P** to add a specific processing to the standard one. The level **10** is not mandatory for ***A** and ***P** types. **segt** represents the Segment code.

  - relative to the level **15** for the access to a Segment for data selection by category.

    The input is as follows:

    ```
    OPE OPERAND                          TY CONDITION
    Function XX   Sub-function YY     Level 15
     N SUB-FCT TITLE                   type SLCT segt CATR
    ```

    Same comments as for the insertion at level **10**.
    CATR is the only possible category.

  - relative to the level **20** for the access to a Segment for data selection by category and elementary service type.

    The input is as follows:

    ```
    OPE OPERAND                        TY CONDITION
    Function XX   Sub-function YY     Level 20
     N SUB-FCT TITLE                type SLCT segt CATR proc
    ```

    Same comments as for the previous insertion. **proc** represents the elementary processing type (**ALIM** for key loading, **CALL** for the call of physical processing, **ERRS** for error processing).

- In the physical accesses (**F80-SEGT-RA**, **F80-SEGT-P** or **F80-SEGT-RN** tags).

☞ Refer to section Insertion Relative to the 'Physical Accesses' Level.

Other Services

User Service

The Pacbench C/S generator allows a Client component to call a special service (called 'User Service'), other than a selection or check/update service (e.g. counter to count the number of customers per day who have issued an order, print service…)

The Business Component retrieves, through the communication area, the name and code of the User Service called by the Client component.

The User Service is associated with a Logical View which is itself associated with a Business Component. If you wish then to implement the same User Service in another Business Component, it is recommended to specify a Business Component dedicated to a User Service. You will thus be able to reuse it.

This method implies the call of the Business Component dedicated to the User Service by a first level Business Component. For more details, refer to section Call of a Business Component by another.

Implementation

To implement a **User Service**, you must:

- declare the Logical View containing the service request issued by the Client component,
- insert the User Service in the Logical View,
- write the User Service in Structured Code.

How to Declare the Logical View

The Business Component must know the **Logical View** for which it implements services.

You declare the Logical View in the Business Component's **`Call of Segments (semi local)`** window.

The use of this window (menus, dialog boxes, etc.) is detailed in a specific chapter of the *VisualAge Pacbase WorkStation* User Interface Guide. In addition, complete documentation for all input fields is found in the *Pacbench C/S – Business Logic & TUI Clients Manual*.

Insertion in the Logical View

A User Service is associated with a Logical View in the **`Procedural Code (-P)`** window of a Business Component. The service is inserted in the Logical View, i.e. in any sub-function which depends on an function **`05*C`** with the code of the Logical View in the **`CONDITION`** column.

This sub-function must be coded as follows:

- a 4-character **code**, the first two identifying the function, the following two being free,
- a **title** (N operator in OPE column),
- a **hierarchical level**: 15,
- the **structure type** (TY column): *C
- the **name of the User Service** preceded by the word USER (CONDITION column)

  This name is limited to 25 characters maximum.

  ☞ TUI applications only:
  In the Client component (occurrence of the C/S Screen entity) the User Service name must match the one set by the Client component in the TECH-SRVUSR field of the communication area.

  Example:

```
 OPE OPERAND                                   TY CONDITION
Function XX                               Level 05
 N FCT TITLE                                   *C VL00
Function XX   Sub-function YY            Level 15
 N SUB-FCT TITLE                               *C USER USERSERVICENAME
```

How to Write the User Service

You write the specific code for the User Service in Structured Code.

👓 For details about inserting specific code, refer to sections General Principles and Important Rules.

Structure of the Generated Code

Working Storage Section

No area specific to the User Service is generated.

Linkage Section

TECH-SRVUSR	User Service name in 25 characters maximum

Procedure Division

For the **'Logical View' block**, the following tag is generated:

```
 10     User service             FVIEW-USER
```

For **logical accesses**, transfers and checks are generated only for the Logical View with which the User Service is associated.

No **physical access** is generated.

Call of a Business Component by another

A Business Component executes the service calls it receives via the message and the associated Logical View. To do so it accesses databases and sends the answer back via the same message.

To execute a service, the Business Component may call another Business Component. This allows Business Components to be specialized and reused as much as possible.

A standard function of Business Logic is the ability to call one Business Component from another Business Component. This allows you to associate several Logical Views called in different Business Components, with a Logical View.

Roughly speaking, the principle is the following:

- a Business Component sends a service request to Logical View 1.
- To implement this service, the first-level Business Component calls a second-level Business Component.
- The second-level Business Component then behaves like a Client component for this service call.
- The service request and answer are conveyed using Logical View 2.

☞ The call of a second-level Business Component is executed according to the service requested by the Client component and the use in reception or display of the second-level Logical View.

For example, if the service issued by the Client component is **`E`** *(check), the second-level Business Component will be called only if the use for Logical View 2 is* **`L`**, **`E`** *or* **`X`**.

Implementation

To implement this service call, you must:

- declare the Logical View 2 which conveys data between the two Business Components,
- identify the second-level Business Component,
- specify the type of service to be executed on Logical View 2.

Open the **`Call of segments (semi local)`** window of the calling Business Component (first-level Business Component) and enter:

- the code of the first Logical View which transmits the service call to the first-level Business Component,
- the call of the second-level Logical View (via the appropriate tree structure) by its code in the first-level Business Component,
- the service call to the second-level Business Component.

You specify the following three elements in the **`Characteristics of the Segment`** dialog box, opened by a double-click on the code of the second level Logical View:

- ♦ VisualAge Pacbase code of Logical View 2 which transmits the service call to the second-level Business Component:
  Its description type is automatically set to **`1`**, whatever value you enter.
  The description of the Logical View's application data will therefore be generated in the Business Component **`DATA DIVISION`**.
- ♦ Second-level Business Component:

* its VisualAge Pacbase code,

* its organization:

X Global call of the Business Component:
The first-level Business Component executes service calls and acts like a Client.
Its function should be considered like a branching point to the second-level Business Components which execute the *full* service requested; the call of the second-level Business Component is only performed *once for all instances*.

2 Instance by instance Business Component call:
The second-level Business Component is called *for each instance*. This allows Business Components to be called as *additional* servers with respect to the first-level Business Component. As a result, a service can call Business Components and data access Segments.

☞ This type of call allows a TUI application to work in folder mode:
The TUI client makes a service call to an aggregated Logical View containing a number of Logical Views. This service is executed by a first-level Business Component whose role is to call – for each instance of the aggregated Logical View – the different Business Components associated with the other Logical Views.

♦ Type of service to be executed:

L read
E check
X check, selection
A selection
M check, update
T check, update, selection
U User Service

Structure of the Generated Code

Working Storage Section

OPER2 Service requested during the call of a second-level Business Component

COMMUNICATION – bc2code
Communication area between the first-level Business Component and the second-level Business Component.
bc2code = code of second-level Business Component

Linkage Section

No specific field for the second-level Business Component is generated but the **TECH-ICATRS** field is loaded with the number of instances retrieved from the second-level Business Component.

Procedure Division

For the **'Logical View' block,** the tags generated depend on of the service executed by the second-level Business Component.

The type of general processing is thus :

CHUP (check / update),

SELC (selection),

USER (User Service).

👓 Refer to the corresponding paragraphs of the executed service.

The second-level Business Component executes the **data transfers** from the Logical View to the Segment or from the Segment to the Logical View, or both directions, according to the service executed. Tags are as follows:

TRDT	TRansfer to "DaTa" in the direction View → Segment
or	
TRVW	TRansfer to "VieW" in the direction Segment → View

👓 Refer to the corresponding paragraphs of the executed service.

For **logical accesses**, the generated tags depend on the service executed by the second-level Business Component.

The type of general processing is thus :

CHCK (check),

UPDT (update),

SLCT (selection)

USRS (User Service).

👓 Refer to the corresponding paragraph of the executed service.

☞ If the use in reception is U (User Service), the call of the logical access functions must be executed by the XT operator. (Refer to *Pacbench Client/Server – Business Logic & TUI Clients Manual*).

For **physical accesses**, tags are as follows:

- F80
- the Segment code in four characters: SEGT
- the access code:

R	Call of the second-level Business Component

- the type of general processing:

ALIM	loading of communication area
CALL	call of Business Component
RETC	Processing at the Business Component return.

```
10                   Call of Business Component           F80-SEGT-R
                  15 Loading of communication area        F80-SEGT-R-ALIM
                  15 Call of Business Component           F80-SEGT-R-CALL
                  15 Return processing                    F80-SEGT-R-RETC
```

Inserting Specific Code

VisualAge Pacbase allows you to implement processes that are not standard processes.

For example, you can call the second-level Business Component only when the order exceeds a given amount or when the order is issued by a special type of customer.

👓 For more details on how to insert specific code, refer to sections General Principles and Important Rules.

Specific code can be inserted:

- In Logical View processing.

  👓 Refer to the corresponding paragraph of the executed service.

- In data transfers.

  👓 Refer to the corresponding paragraph of the executed service.

- In logical accesses

  👓 Refer to the corresponding paragraph of the executed service.

- In the physical accesses

  If the communication between the two Business Components is not as you want it to be or if you want to add complements in the Business Component calls, you can write your own processing in Structured Code.

  👓 Refer to section Insertion Relative to the 'Physical Accesses' Level.

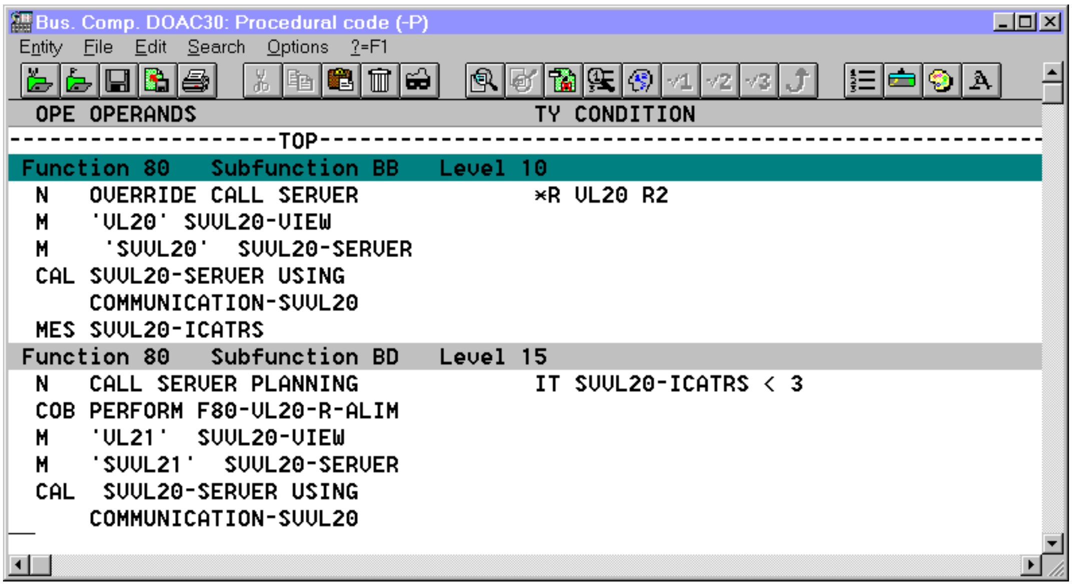
  - relative to the level **10** (**F80-SEGT-R** tag).
  - relative to the loading process of the communication area at level 15 (**F80-SEGT-R-ALIM** tag):

    To replace this access, enter the Segment code in the **CONDITON** column followed by **R1**.
  - relative to the Business Component call at level **15** (**F80-SEGT-R-CALL** tag):

    To replace this access, , enter the Segment code in the **CONDITON** column followed by **R2**.
  - relative to the processing at the Business Component return at level 15 (**F80-SEGT-R-RETC** tag):

    To replace this access, enter the Segment code in the **CONDITON** column followed by **R3**.

Here is an example of specific code insertion which specifies the call of the second-level Business Component SVVL20. The Business Component SVVL10 is called in standard processing. The Business Component SVVL20 is called only when the number of available items is lower than 3. You must insert specific code at level 15 (F80-SEGT-R-CALL tag) where the call of this Business Component is managed.



```
 OPE OPERANDS                                TY CONDITION
--------------------TOP------------------------------------------------------
Function 80   Subfunction BB   Level 10
 N   OVERRIDE CALL SERVER                    *R VL20 R2
 M   'VL20' SVVL20-VIEW
 M    'SVVL20'  SVVL20-SERVER
 CAL SVVL20-SERVER USING
     COMMUNICATION-SVVL20
 MES SVVL20-ICATRS
Function 80   Subfunction BD   Level 15
 N   CALL SERVER PLANNING                    IT SVVL20-ICATRS < 3
 COB PERFORM F80-VL20-R-ALIM
 M   'VL21'  SVVL20-VIEW
 M   'SVVL21'  SVVL20-SERVER
 CAL  SVVL20-SERVER USING
     COMMUNICATION-SVVL20
```

Error Handling

The retrieval of errors during the call of a second-level Business Component is automatically executed in the following area:

- the Logical View code in four characters: **FVIEW**
- the transfer of errors:
  TRER Transfer of the second-level Business Component errors
- the code of the Segment where the error is detected, in four characters: **SEGT**

```
10      Transfer data errors                  FVIEW-TRER
   15   Transfer View of Server call          FVIEW-TRER-SEGT
```

Lock / Unlock Service

☞ This service is not available for simple or TUI development modes.

The Lock/Unlock service allows you to prevent the update for a set of data to avoid simultaneous update, i.e to avoid inconsistency.

You write the processing associated with this service in Structured Code.

Implementation

To implement a **lock / unlock** service, you must:

- set the lock option,
- declare the Logical View containing the service request issued by the Client component,
- insert the service in the elementary Business Component associated with the Folder *root* node or in the Business Component if no Folder has been implemented,
- write the service in Structured Code.

Lock Option

This option must be specified in the Folder Dialogue or the Folder **Definition** screen.

The implementation of this option is documented in section Lock Option in Chapter 6: Folder and Folder Views (graphic applications).

If you specify no Folder (single-view development), you must set the **LOCKMODE** option in the **Generation Parameters** screen of the Dialogue (or Business Component). Refer to section Additional Options for Single-View Development.

How to Declare the Logical View

The Business Component must know the **Logical View** for which it implements services.

You declare the Logical View in the Business Component's **Call of Segments (semi local)** window.

The use of this window (menus, dialog boxes, etc.) is detailed in a specific chapter of the *VisualAge Pacbase WorkStation User Interface Guide*. In addition, complete documentation for all input fields is found in the *Pacbench C/S – Business Logic & TUI Clients Manual*.

Insertion in the Logical View

You must insert the lock and unlock service in the specific processing lines of the Business Component.

For more details on how to insert specific code, refer to sections General Principles and Important Rules.

The processing must be written in any sub-function which depends on an **05*C** function with the code of the root Logical View in the **CONDITION** column.

- Each sub-function must be coded as follows:
  - a 4-character **code**, the first two identifying the function, the following two being free,
  - a **title** (**N** operator in **OPE** column),
  - a **hierarchical level**: necessarily higher than 10,
  - the **structure type** (**TY** column):
    ***P** after (post) standard processing:
    - * just before the ending tag, if the hierarchical level is higher than the standard one,
    - * just after the ending tag if the hierarchical level is equal to the standard one.
  - the **service** name (**CONDITION** column):
    LOCK
    UNLK

Example:

```
 OPE OPERAND                          TY CONDITION
Function XX                       Level 05
```

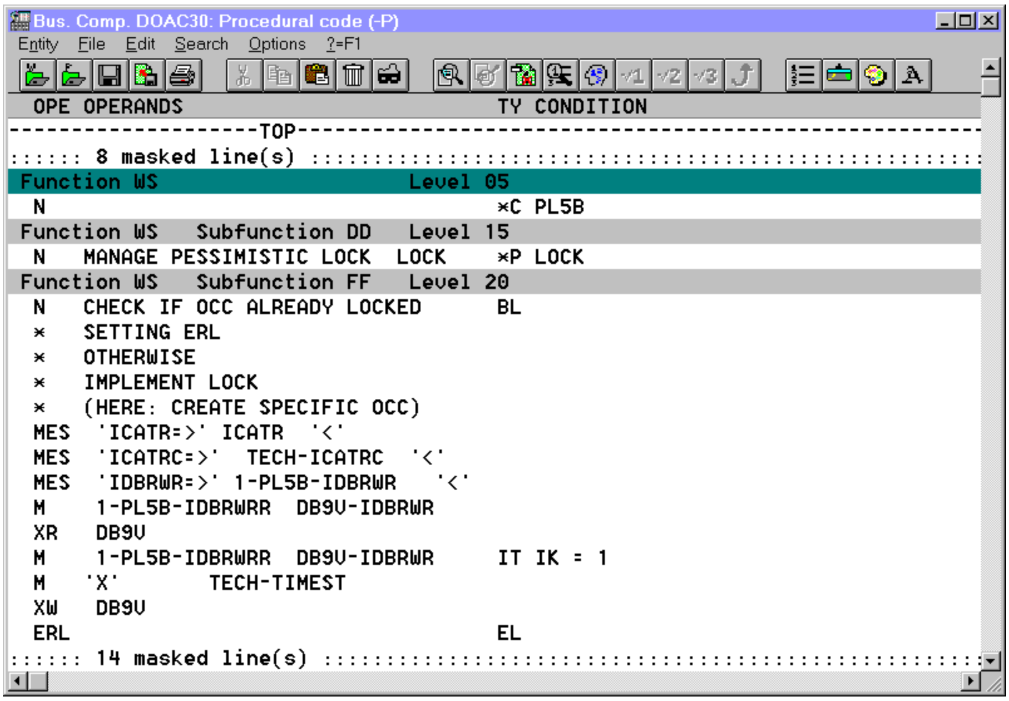
```
 N   TITLE FCT                        *C VL00
Function XX   Sub-function YY    Level 15
 N SUB-FCT TITLE                      *P LOCK
```

How to Write Lock / Unlock Service

You write the specific code for the lock/unlock service in **Structured Code**

Fore more details on how to insert specific code, refer to sections General Principles and Important Rules.

Here is an example of specific code written for a lock service


Bus. Comp. DOAC30: Procedural code (-P)
Entity File Edit Search Options ?=F1

```
 OPE OPERANDS                           TY CONDITION
--------------------TOP---------------------------------------------------
:::::: 8 masked line(s) :::::::::::::::::::::::::::::::::::::::::::::::::
Function WS                      Level 05
 N                                      *C PL5B
Function WS   Subfunction DD    Level 15
 N   MANAGE PESSIMISTIC LOCK  LOCK     *P LOCK
Function WS   Subfunction FF    Level 20
 N   CHECK IF OCC ALREADY LOCKED       BL
 *   SETTING ERL
 *   OTHERWISE
 *   IMPLEMENT LOCK
 *   (HERE: CREATE SPECIFIC OCC)
 MES  'ICATR=>' ICATR  '<'
 MES  'ICATRC=>'  TECH-ICATRC  '<'
 MES  'IDBRWR=>' 1-PL5B-IDBRWR   '<'
 M    1-PL5B-IDBRWRR  DB9V-IDBRWR
 XR   DB9V
 M    1-PL5B-IDBRWRR  DB9V-IDBRWR     IT IK = 1
 M   'X'        TECH-TIMEST
 XW   DB9V
 ERL                                    EL
:::::: 14 masked line(s) ::::::::::::::::::::::::::::::::::::::::::::::::
```

Structure of the Generated Code

Working Storage Section

No area specific to the (un)lock service is generated.

Linkage Section

No area specific to the (un)lock service is generated.

Procedure Division

For the **'Logical View' block**, only the following tag are generated:

```
 10     Lock                  FVIEW-LOCK
 10     Unlock                FVIEW-UNLK
```

For **logical** and **physical accesses** , no area specific to the (un)lock service is generated.

Error Handling

Error handling must be managed in specific code using the **ERL** operator.

This operator allows to throw an error when a **LOCK** or **UNLOCK** service is issued by the graphic client, respectively for an instance that is already locked or unlocked. This operator generates **MOVE 'L' TO TECH-IERRU**.

Initialization/Termination Business Component

Introduction

An *Initialization/Termination Business Component* (**I/T** *Component*) is used to implement specific procedures before and after the execution of a request associated with a Folder.

☞ This functionality is then only available for graphic applications using the development of a Folder.

An **I/T** Component is called by the Services Manager before the first call of Business Component associated with the request to be processed and after the last executed Business Component.

An **I/T** Component is available either for an Initialization or for a Termination process (see the implementation details below).

For an initialization process, the only data that can be processed in input is that sent by the Client component via the User Buffer.

An **I/T** Component is used to execute services associated with all the Business Components of a Folder.

Therefore, the generation of an **I/T** Component contains the access functions and **PERFORM**s corresponding to the services associated with all the Business Components of a Folder.

Each service associated with a node uses specific parameters in input and produces results defined in the **WORKING-STORAGE-SECTION** of the **I/T** Component.

How to use a Buffer

An **I/T** Component systematically retrieves the User Buffer(s) specified at the Folder (Business Components) level.

External Business Component called by I/T Component and User Buffer

- If the Business Components associated with the root and depending nodes of the Folder do not use a User Buffer:
    - the external Business Component called by the **I/T** Component cannot use a User Buffer.
- If the Business Components associated with the Folder use a User Buffer :
    - the external Business Component called by an **I/T** Component does not have a User Buffer
    - the external Business Component called by an **I/T** Component must use a User Buffer associated with one of the Folder nodes.

External Business Component called by I/T Component and Server Buffer

- If the Business components of the Folder use a Server Buffer, the external Business Component called by an **I/T** Component must have the same Server Buffer.

- If the Business components of the Folder do not use a Server Buffer, the external Business Component called by an **I/T** Component cannot use a Server Buffer.

Implementation

Definition of the I/T Component

The Definition of a Business Component consists in creating an occurrence of the Business Component entity (6-character **code,** the first two characters being identical to those of the Dialogue code, previously defined).

In the **Definition** window, enter :

- the **name**
- optionally, one or more **keywords**
- the **IT** type
- the **generation variants**

  transaction *code: unused*
- the **external name** on the **Program** line (default = code of the **I/T** Component).
- the in front/in back commands (**CCF**/**CCB**) on the **Program** line

  Complement line : unused

Options of the I/T Component

<table>
<tr><td>ACCESERR</td><td colspan="3">Maximum number of accesses to Segment error that can be returned to the Client, on 3 characters. Default Value =001
NB : the value specified here must be identical to that given to the ACCESERR option for all Business Components (levels 1 and 2) of the Folder</td></tr>
<tr><td rowspan="10">CALLTYPE</td><td colspan="3">NB : the value specified here must be identical to that given to the CALLTYPE option for all the Business Components (levels 1 and 2) of the Folder</td></tr>
<tr><td><b>Target</b></td><td><b>GUI</b></td><td><b>TUI</b></td></tr>
<tr><td>CICS</td><td>LINK*
CALL</td><td>LINK*
CALL</td></tr>
<tr><td>DPS7</td><td>CALL*</td><td>CALL*</td></tr>
<tr><td>DPS8</td><td>LINK*
CALL</td><td>LINK*
CALL</td></tr>
<tr><td>IMS</td><td>CALL*</td><td>CALL*
CHNG</td></tr>
<tr><td>MICROFOCUS</td><td>CALL*</td><td>CALL*</td></tr>
<tr><td>TANDEM PATHWAY</td><td>PATHSEND</td><td></td></tr>
<tr><td>TUXEDO</td><td colspan="2">The CALLTYPE value is automatically that of the Business Components associated with the Folder, i.e. TPCALL or CALL.
The setting of this option is then not required for this target[2].</td></tr>
<tr><td colspan="3">* : default value</td></tr>
<tr><td>CHANGE</td><td colspan="3">Option specific to TUXEDO :
Default value : NO Call of Business Component with the TPNOCHANGE parameter of the CALL
If CHANGE=YES : call of Business Component with the TPCHANGE parameter of the CALL</td></tr>
<tr><td>DATAERR</td><td colspan="3">Maximum number of errors on the Logical View Data Elements that can be sent back to the Client, 2 characters
Default value = 01
NB : The value specified here must be identical to that given to the DATAERR option for all the Business Components associated with the Folder</td></tr>
<tr><td>PROCESS</td><td colspan="3">Available for TANDEM PATHWAY only.
Name of the Process when calling the Initialization Component.
15 characters maximum without space. There is no lowercase/uppercase conversion.</td></tr>
<tr><td>TRAN</td><td colspan="3">Option specific to TUXEDO
Default value : TRAN=YES : Use of the transactional mode for the TUXEDO variant.
TRAN=NO : No use of the transactional mode</td></tr>
</table>

Implementing the I/T Component at the Folder Level

The INITSERV option must be set in the Generation Parameters window of the Folder (O-type line).

INITSERV	INITSERV = VisualAge Pacbase code of the Initialization/Termination Business Component (I/T Component) This option must be aligned on the left.

[2] With the RC variant only, you can however modify this option dynamically by modifying the value of the V-xxxx-TYCALL indicator, initialized by the CALLTYPE option of the Business Component (level 1) : C if CALLTYPE=CALL, else T.

Call of an External Business Component

The services of an external Business Component process, as for the Business Components associated with the Folder nodes, only one Logical View instance.

☞ Each Logical View used can only be called once and must not be already used in one of the Folder Business Components.

The call of an external Business Component is made in the **Call of Segments (semi local)** window of the **I/T** Component, in the **List of Unused Segments** dialog box opened via the **Unused Segments** choice in the **View** menu.

You only specify the following information :

Code	Code of the Logical View in the external Business Component
External name	Code of the external Business Component in the Dictionary
Code in base	Code of the Logical View in the Dictionary.
Organization 2	Call of Business Component (instance by instance)
Type of description 1	required value

Structure of the Generated Code

Working Storage Section

Each service associated with a node uses specific parameters before the call of the Business Component associated with the node and produces results in return.

These parameters are specified in the **WORKING-STORAGE-SECTION** of the **I/T** Component. Only the fields that can be handled or that are useful are documented in the table below.

Summary table :

Parameters	**Field name**	**Before call**	**After call**
Indicator of logical error on Business Component	SERV-ERR		yes
Description of node :	V-xxxx		
Operation code in update	V-xxxx-OPERM	yes	
Indicator of data check	V-xxxx-IDATAC	yes	
Code of the extraction method	V-xxxx-EXTNAM	yes	
Code of the User Service	V-xxxx-SRVUSR	yes	
Indicator system error	V-xxxx-IERRC		yes
Indicator end of selection or absence of instance	V-xxxx-IERRS		yes
Indicator logical or user error	V-xxxx-IERRU		yes
Indicator selected instance (1 or 0)	V-xxxx-ICATRS		yes
Description of Logical View	xxxx	yes	yes
Description of Logical View presence indicator	CH-xxxx	yes	yes

SERV-ERR indicator of logical error on a Business Component. This field contains the **E value**, as soon as a logical error has been detected on a Business Component.

- A **V-xxxx** structure is generated for each Folder node

  V-xxxx-OPERM type of service requested for the update. By default, this is a service without selection. A service with selection may be requested in specific code by entering the **T** value. This field is generated only if the Business Component includes an update service..

  V-xxxx-IDATAC check indicator of the Logical View's data in the Business Component. By default, it is set to 'without check'. A check may be requested in specific code by entering the ' ' value.

  V-xxxx-EXTNAM code of the extraction method. By default, it is set to 'no extraction method'. An extraction method may be positioned by inserting specific code.

  V-xxxx-SRVUSR code of the User Service. By default, it is set to 'no User Service'. By default, a User Service may be positioned by inserting specific code.

  V-xxxx-IERRC indicator of system error detected in the Business Component corresponding to the node.

  V-xxxx-IERRS indicator of end of selection (**L**) or absence of instance (**C**) positioned by the Business Component corresponding to the node.
  If in the Business Component, the selection is performed on non relational data, the indicator of end of selection is positioned *at the same time* as the reading of the last recording.
  If the selection is performed on relational databases, the indicator of end of selection is positioned *during the reading following the reading of the last recording*.

  V-xxxx-IERRU indicator of logical or user error detected by the Business Component corresponding to the node.

  V-xxxx-ICATRS number of instance selected by the Business Component during a selection or User Service request (**1** or **0**).

- A description field is generated for each Logical View
  - CAUTION : the field code is that of the node and not that of the Logical View.
  - the description corresponds to the description of an instance and is not 'occursed' by the maximum number of instances of the Logical View.

Example :

```
01          NO10.                             CLINIT
 10       1-NO10-CLINUM PICTURE S9(8)         CLINIT
                 SIGN IS LEADING SEPARATE.    CLINIT
 10       1-NO10-CL1NAM PICTURE X(30).        CLINIT
 10        NO10-CLINUM PICTURE S9(8)          CLINIT
                 SIGN IS LEADING SEPARATE.    CLINIT
 10        NO10-CLINAM PICTURE X(30).         CLINIT
 10        NO10-CMPANY PICTURE X(50).         CLINIT
 10        NO10-STREET PICTURE X(40).         CLINIT
 10        NO10-TOWN   PICTURE X(20).         CLINIT
 10        NO10-ZIPCDE PICTURE X(5).          CLINIT
 10        NO10-CLITYP PICTURE X.             CLINIT
```

- One description field of presence indicator per Logical View

This field is generated if the **CHECKSER** or **VECTPRES** option is coded in the corresponding Business Component.

Example :

```
01        CH-NO10 VALUE ALL "N".              CLINIT
      15      CH-NO10-CLINUM PICTURE X.        CLINIT
      15      CH-NO10-CLINAM PICTURE X.        CLINIT
      15      CH-NO10-CMPANY PICTURE X.        CLINIT
      15      CH-NO10-STREET PICTURE X.        CLINIT
      15      CH-NO10-TOWN   PICTURE X.        CLINIT
      15      CH-NO10-ZIPCDE PICTURE X.        CLINIT
      15      CH-NO10-TYPCLI PICTURE X.        CLINIT
```

  - CAUTION : the field code is that of the node and not that of the Logical View.
  - The description corresponds to that of an instance and is not 'occursed' by the maximum number of instances of the Logical View.
  - The field is initialized to **N**: no data check by default. To check a data item, the ' 'value must be set for the field concerned in specific code. To indicate that the field is not present, the **P value** must be entered.

Linkage Section

- **TECH-BUFFER** technical field

  TECH-IERRU when the **I/T** Component is called in termination, this field contains the indicator of serious error detected in one of the Business Components processed previously. This allows to know if an error has been positioned during the processing of the request.

  TECH-IERRC when the **I/T** Component is called in termination, this field contains the indicator of logical (**C**) or user (**U**) error detected in one of the Business Components processed previously. This allows to know if an error has been positioned during the processing of the request.

  TECH-REQST contains the **I** value if the **I/T** Component is called for initialization, the **T** value if it is called for termination.

- **Server Buffer** (if the Business Components do use one)

Example :

```
03           BS01.                         *00045
  10         BS01-MESSA PICTURE X(100).    *00045
```

- **User Buffer** (if the Business Components do use one)

The first User Buffer generated is the Folder's.

Example :

```
03           BU01.                            *00050
  10         BU01-CLINUM PICTURE S9(8)        *00050
                  SIGN IS LEADING SEPARATE.   *00050
  10         BU01-MESSA  PICTURE X(75).       *00050
  10         BU01-CMPT   PICTURE 9(3).        *00050
  10         BU01-FIL14  PICTURE X(14).       *00050
```

If need be, User Buffers – different and associated with reference nodes – are generated in **REDEFINES** of the Folder User Buffer (they must have the same length).

Example :

```
03            BU02 REDEFINES BU01.               *00050
  10          BU02-PROCOD PICTURE X(5).          *00050
  10          BU02-FIL96 PICTURE X(96).          *00050
```

- Error field in selection

Example :

```
03   SELT-BUFFER.                                *00060
  10 SELT-SEGCOD  PICTURE X(4).                  *00060
  10 SELT-SEGERR  PICTURE X(4).                  *00060
  10 SELT-SEGTYP  PICTURE X.                     *00060
  10 SELT-LIBRA   PICTURE XXX.                   *00060
  10 SELT-SERVER  PICTURE X(6).                  *00060
```

- Free field for the storage of errors detected in the **I/T** Component specific code

Example :

```
03  ERR-BUFFER                                            *00080
  10 V-ERR-BUFSEG OCCURS 003                              *00080
             15 V-ERR-SEGCOD PICTURE XXXX                 *00080
             15 V-ERR-SEGERR PICTURE XXXX                 *00080
             15 V-ERR-SEGTYP PICTURE X.                   *00080
             15 V-ERR-ICATR  PICTURE 9(4)                 *00080
             15 V-ERR-LIBRA  PICTURE X(3)                 *00080
             15 V-ERR-SERVER PICTURE X(6)                 *00080
  10 ERR-DATA OCCURS 002.                                 *00080
             15         EE-DATA-LIBRA   PICTURE XXX.      *00080
             15         EE-DATA-SERVER  PICTURE X(6).     *00080
             15         EE-DATA-VIEW    PICTURE X(4).     *00080
             15         EE-DATA-DATCOD  PICTURE X(6).     *00080
             15         EE-DATA-DATERR  PICTURE X.        *00080
             15         EE-DATA-DATTYP  PICTURE X.        *00080
             15         EE-DATA-ICATR   PICTURE 9(4).     *00080
```

Procedure Division

- Access Functions

Each service of a Business Component is associated with a sub-function of function **80** capable of managing one instance of a node at a time. Thus, for a given Business Component, the different access sub-functions generated in the **I/T** Component are the following :

F80-NodeCode-R	Direct reading of an instance
F80-NodeCode-RU	Execution of a User Service
F80-NodeCode-P	Opening of a cursor, first reading
F80-NodeCode-RN	Next Reading of an instance

`F80-NodeCode-W`	Writing of an instance
`F80-NodeCode-RW`	Modification of an instance
`F80-NodeCode-D`	Deletion of an instance
`F80-NodeCode-EB`	Closing of a cursor
`F80-NodeCode-RA`	Common Initializations and loading of the User Buffer if present for the node processed
`F80-NodeCode-UN`	Initializations at the Business Component return and loading from the User Buffer if present for the node processed.

Each Business Component of a Folder is identified in an **`I/T`** Component by the code of the node it manages.

These sub-functions handle :

- the initialization of the communication area of the Business Component to call from its characteristics and the input parameters provided,
- the call of the Business Component,
- the loading of the results expected after this call. If a serious error is detected, the **`I/T`** Component immediately forwards it to the Services Manager and the transaction is cancelled before the answer is sent back to the Client component.

- **`PERFORMs`**

The following sub-functions of Function **`81`** are generated :

`F81AL.` This function performs common initializations, independent of nodes, before the call of the Business Component. The loading of the Server Buffer is generated if the Folder has a User Buffer.

`F81CA.` This function performs the call of the Business Component (syntax depending on the variant and the **`TYCALL`** option indicated at the **`I/T`** Component level), common initializations independent of the nodes at the Business Component return, in particular related to the handling of errors. The loading from the Server Buffer is generated if the Folder has a Server Buffer.

`F81EE.` This function is not called automatically. It allows you to forward access or user errors – detected in the Business Component – to the Services Manager and Client component.

Error Handling

Logical access errors, check errors of Logical Views data and user errors – detected by the Business Component – are not automatically forwarded to the Services Manager and the Client component.

You decide if these errors have to be visible on the client side by calling the generated functions managing their sending.

You can use the **ERU** operator in the specific code cinematic to forward a user error on to the Business Component. In this case, this error is considered by the Client component as a general error on the request and its context cannot be restored.

Chapter 4: Error Handling

In the Business Component, you specify processing, i.e. services associated with a Logical View.

As processing is being executed, the Business Component may detect errors. In this case, it must send all the error information in its reply message to the Client, so that the Client can take the appropriate action.

Principles

When a Business Component detects an error, a `ROLLBACK` is immediately executed– before sending back the reply containing the errors – with a return to the Client part, without calling the next Business Component.

In order to avoid a systematic exchange between the Client and the Business Component for *each* error, the Services Manager saves *all* the errors detected by the Business Component, accesses the Error Message Server and returns the corresponding explicit error messages.

The Services Manager sends back to the Client – via the Communications Monitor – the application errors detected by the Business Component and inserts them in the communication area for Client processing.

To implement error message handling, you must:

- Set options in the Folder or Business Component Dialogue,
- Describe – in each Business Component and Logical View – the error messages associated with the codes of errors processed via specific code,
- Create and describe the Error Message Server,
- Generate the error message file.

Options to Be Specified

A Business Component may detect errors as a procedure is being executed.

The number of errors it can detect before sending control to the calling program is defined by the `ACCESERR` and `DATAERR` options, one for each error type.

You enter these options on `O`-type lines, at the top of the `Generation Parameters` screen, as follows: `OPTION=option value` (with no space before and after `=`).

Several options, separated by at least one blank, may be entered on the same line.

No check is executed when you enter the options, but if they do not correspond to standard options, messages will be displayed in the generation report..

- **Options of the Business Component Dialogue**

ACCESERR	Maximum number of errors on Segment access that can be returned to the Client. 3-character value Default value = 001
DATAERR	Maximum number of errors on the Logical View's Data Elements that can be returned to the Client. 2-character value Default value = 01

☞ These two options are the only ones required for the development of a TUI application.

- **Options of the Folder (Dialogue)**

ERRLAB	This option allows you to specify how the errors are to be returned. By default, **ERRLAB=YES** : the Communications Monitor retrieves the key, the gravity and the error message returned by a Business Component, before sending its answer to the Client Component. Conversely, if you set this option to **NO**, only the error key will be returned. In this case, the client application has to retrieve a local error message.

Options Specific to Single-View Development

The options contained in the following table must be entered in the `Generation Parameters` screen of the *Business Component Dialogue*.

ACCESERR	Maximum number of errors on Segment access that can be returned to the Client. 3-character value Default value = 001
DATAERR	Maximum number of errors on the Logical View's Data Elements that can be returned to the Client. 2-character value Default value = 01

ERRLAB	This option allows you to specify how the errors are to be returned. By default, **ERRLAB=YES** : the Communications Monitor retrieves the key, the gravity and the error message returned by a Business Component, before sending its answer to the Client Component. Conversely, if you set this option to **NO**, only the error key will be returned. In this case, the client application has to retrieve a local error message.
ERRSERV	VisualAge Pacbase code of the Error Message Server. **This option is required**.

Description of Error Messages

There are two types of errors:

- **Standard errors:**

  Standard errors are detected after standard accesses to Segments or checks on Logical View data.

  👓 You can modify the messages associated with standard errors. For details, refer to the *Business Logic & TUI Clients Reference Manual, Chapter Error Messages – Help Function (TUI), Subchapter Error Messages- Introduction, section Business Component: Overriding Automatic Error Messages.*

- **Specific errors:**

  Specific errors correspond to errors set in the Business Component via the **ERU** (user error), **ERR** (error on Data Element) or **ERL** (logical lock or unlock error) operators.

✍ The writing of specific code (in Structured Code) is documented in *Subchapter* Inserting Specific Code. For details on these operators, refer to the *Business Logic & TUI Clients Reference Manual, Chapter Business Component, Subchapter Writing Procedural Code, section Operators Used by Pacbench Client/Server, paragraph Operators for Error Positioning*.

You enter the messages associated with these errors in:

- the **Business Component** for the errors specified by **ERU** or **ERL**,
- the **Logical View** for the errors specified by **ERR**.

Messages Associated with the Business Component

In the **Error messages** screen of the Business Component, you describe the messages of the **user errors** via **U**-type lines.

In the **DESCRIPTION** field, you enter:

- the **error code** in 4 characters (columns 1 to 4) as used in the specific procedures,
- the **error gravity** in 1 character (column 5):
  E serious error,
  W warning,
- the **message** associated with this error (beginning in column 6).

Messages Associated with the Logical View's Data Elements

You can define errors in a Business Component for the Logical View data. You must associate messages with these errors.

You declare these messages in the **Logical View**. For each Data Element concerned by an error, a message must be declared in the **Error messages** screen associated with this Data Element.

You describe these error messages on **U**-type lines.

In the **DESCRIPTION** field, you enter:

- the **error code** in 1 character (column 4) as used in specific code,
- the **error gravity** in 1 character (column 5):
  serious error,
  E serious error,
  W warning,
- the **message** associated with this error (beginning in column 6).

The Error Message Server

You must create an Error Message Server to access the error message file.

You specify the Error Message Server in the following two windows:

- **Definition**,
- **Call of Segments (semi local).**

Definition

You create the Error Message Server by entering its Definition.

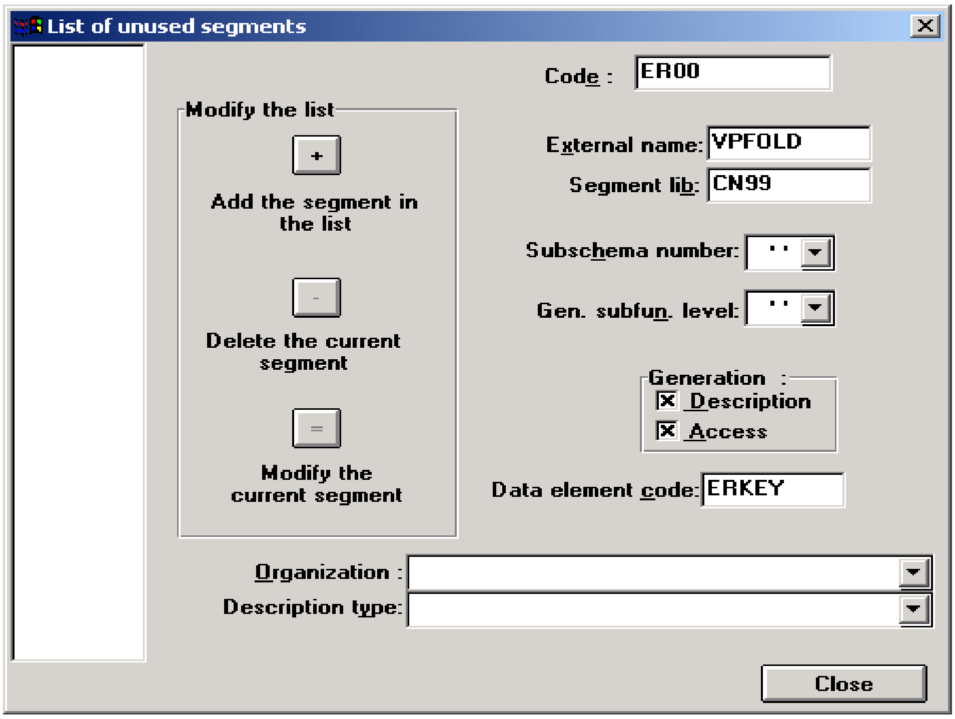
This type of Server must have a specific type:

E: Error Message Server.

Access to the Error Message File

You code the access to the error message file from the Call of Segments (semi local) window in the List of Unused Segments dialog box opened via the Unused Segments choice in the View menu. You specify the following information for the ER00 Segment code:

External name	VisualAge Pacbase code of the DB block if H organization or file logical name (DDNAME)
Segment lib	Segment code in the Dictionary[3].
Key Data Element	ERKEY
Organization	H (for a relational table) and V for a VSAM file
Description type	1


List of unused segments
Modify the list
+
Add the segment in the list
-
Delete the current segment
=
Modify the current segment
Close

Code : ER00

External name: VPFOLD

Segment lib: CN99

Subschema number: ' '

Gen. subfun. level: ' '

Generation :
☒ Description
☒ Access

Data element code: ERKEY

Organization :

Description type:

To process errors, four variables will be generated in the WORKING-STORAGE SECTION of the Business Component:

- IER number of errors on Segment access, initialized by the ACCESERR option
- IED number of errors on Data Element, initialized by the DATAERR option
- K50L work index of the number of errors on Segment access

[3] For the description of this particular Segment, see next page.

- **K50D** work index of the number of errors on Data Element

👓 For more information on how to generate the Error Message Server, refer to *Subchapter* Generation.

The Error Message File : description and generation

An application's error messages are generated from information stored in the Dictionary.

Description

For a graphic application, the structure of the Error Message File must be described via a Segment occurrence. This description follows different and particular rules according to the file organization type.

👓 For complete information on how to describe the Error Message File, consult the *Business Logic & Clients TUI manual*, Chapter *Business Component*, *Subchapter Call of Segments in the Business Component (-CS)*.

Generation

You must generate these messages in a file via the **GEC** command of the **GPRT** procedure in option **C1**, by indicating the Dialogue code of your application, therefore including all the attached Business Components.

The file that is generated is a sequential file from which you create the error message file for your application, with the appropriate organization (relational table, VSAM file, etc.).

☞ If the suffix of a Business Component code is entered on the continuation line of the **GEC** command, error messages are generated for this Business Component only.

The error message indexed file includes 100-character records whose structure is:

- an access key (**ERKEY**, 29 characters),
- a gravity code (1 character: **E** for error, **W** for warning),
- the message itself.

👓 The structure of the error message file is described in the *Business Logic & TUI Clients Reference Manual, Chapter Error Messages – Help Function (TUI), Subchapter Error Messages Generation and/or Printing, section Description of the Error Message File*.

Chapter 5: Communications Monitor (graphic applications)

✍ The operating principles of the Communications Monitor are presented in the Developer's Documentation / eBusiness and Pacbench/CS Applications: **Concepts & Architectures**.

Definition

A Communications Monitor is defined by creating an occurrence of the **Business Component** entity. It is recommended that the Communications Monitor be attached to the application's Business Components Dialogue.

In the `Definition` window, you enter:

- the **name** of the Communications Monitor (required)
- one or more **keywords** (optional)
- the **type** `FM` (required)

After transmission, the `Definition` window displays the following fields:

- the message length [x 1024 characters]

  The maximum physical size of a message between Client and Server – in kilobytes - between `2` and `32` K.

  This value depends on the capacity allowed by the network administration of the application's execution environment. Default = `24`

  The size of the message sent on the network corresponds to the useful size of its content.

  ☞ In case you have generated the Communications Monitor with a former version of Pacbench/CS, refer to information concerning compatibility of versions, delivered with the installation folders of the current version.

- sending of error labels option

`Y`	sends error labels (default, Dialogue)
`N`	sends error keys only
`blank`	uses the option selected for the Dialogue

- the generation **variants**
- the **commands in front/in back**
- the **external name** (default: Communications Monitor code)
- the communication type (required):

ECI NON EXTEND	`ECINOEXT`
SNACPIC or MSCPIC	`CPIC`
TUXEDO XA	`XA`
TUXEDO NON XA	`NONXA`
TCP-IP	`SOCKET`
TCIS	`TCIS`
MQSERIES	`MQSERIES`
XCP2/CPI-C (via Tuxedo/Host Connect)	`XCP2`
Local Communication (default value)	`LOCAL`
TCP/IP Access TDS (for GCOS7)	`TCPTDS`
MQBRIDGE	`MQBRIDGE`

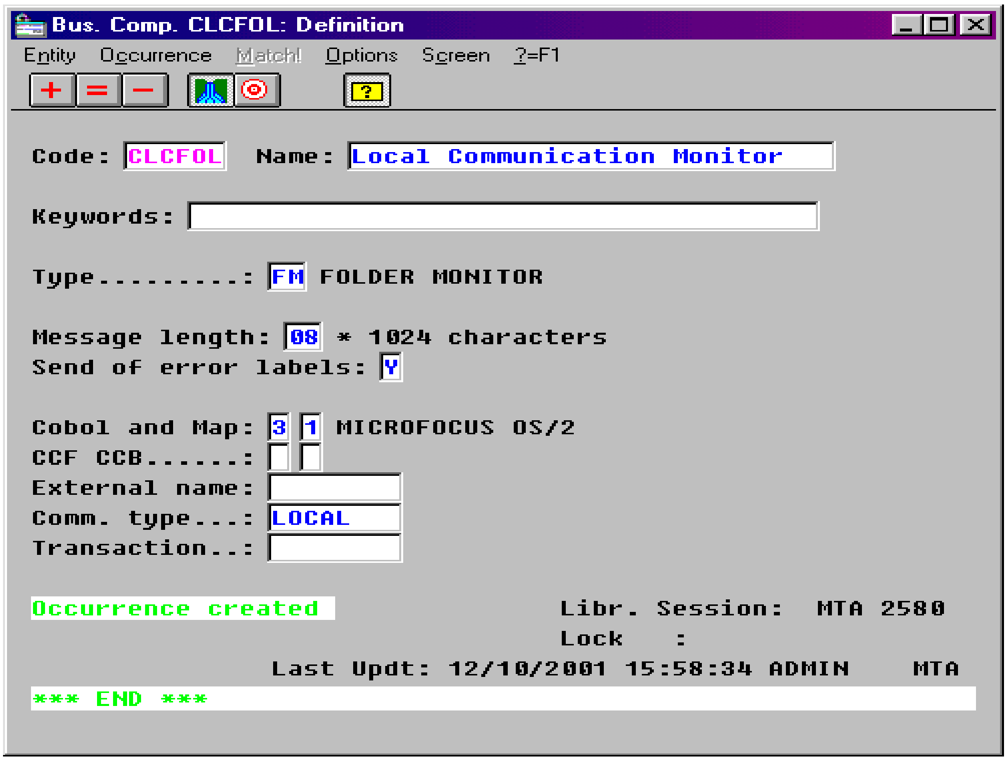
GTEA (for GCOS8) **GTEA**

The choice of the Communication type depends on the generated COBOL variant and possibly on the Transactional Monitor variant. Consult the summary table of compatibilities in the *Developer's Documentation / eBusiness and Pacbench/CS Applications: **Concepts & Architectures***, Chapter *Execution Environments*.

- the **transaction code**

If you press **F1** while the cursor is in an input field, on-line help is displayed; it contains a list of all possible values. If you double-click on the desired value, it will be automatically entered in the field.

*Example of the **Definition** of a Communications Monitor (**FM** type):*


Bus. Comp. CLCFOL: Definition
Entity Occurrence Match! Options Screen ?=F1

```
Code: CLCFOL   Name: Local Communication Monitor

Keywords:

Type.........: FM FOLDER MONITOR

Message length: 08 * 1024 characters
Send of error labels: Y

Cobol and Map: 3 1 MICROFOCUS OS/2
CCF CCB......:
External name:
Comm. type...: LOCAL
Transaction..:

Occurrence created              Libr. Session:  MTA 2580
                                Lock    :
               Last Updt: 12/10/2001 15:58:34 ADMIN    MTA
*** END ***
```

Setting Generation Options

A number of options are needed when generating a Communications Monitor.

You enter these options at the top of the **Generation Parameters** window of the Communications Monitor, on **O**-type lines as follows: **OPTION=option value** (with no space before and after **=**).

Several options, separated by a blank, may be entered on each line.

No check is performed when you enter the generation options. But if the options you enter are not standard options, messages will be displayed in the generation report.

If the Communications Monitor belongs to your application's Business Component Dialogue, you can set these options directly in the Dialogue. They will be automatically used during the generation of the Communications Monitor.

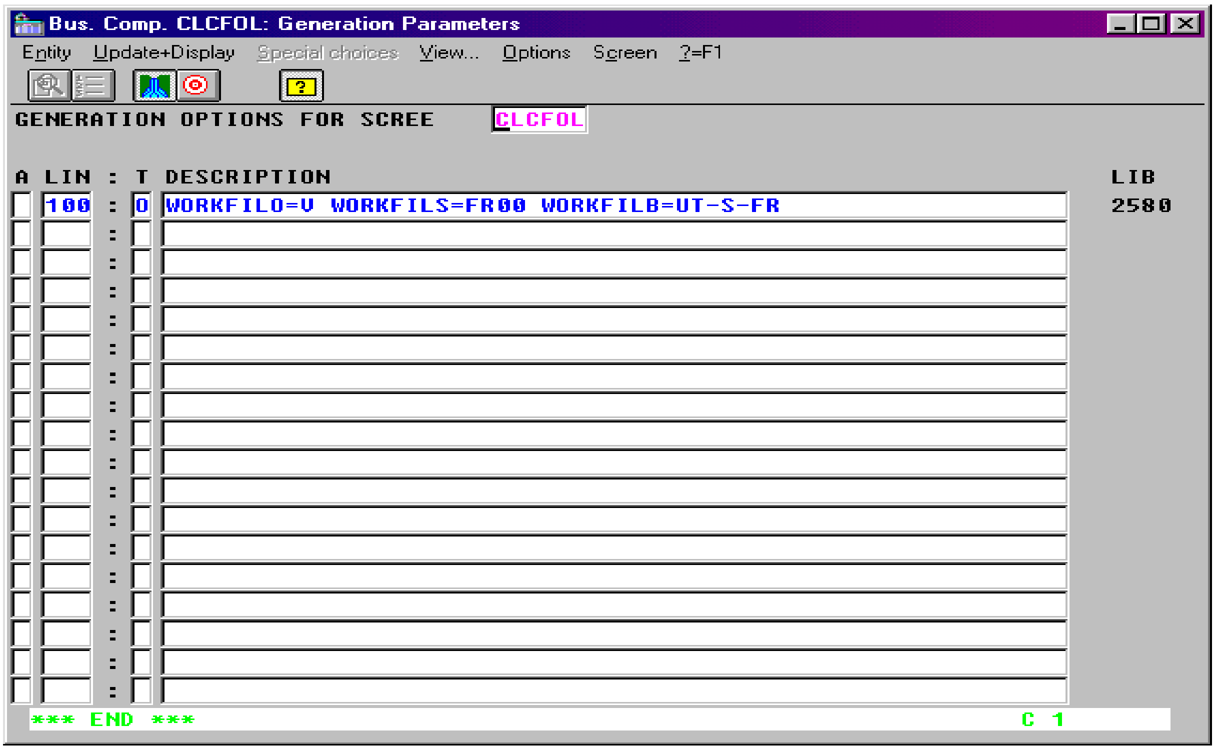
<table>
<tr><td>BASE</td><td colspan="3">VisualAge Pacbase code for the SQL Database.
As long as a relational table belongs to the Folder, this option is required (even if it is a work file).
BASE=THREAD: Value reserved for the UNISYS-2200 to indicate an SFS organization</td></tr>
<tr><td>BASELOC</td><td colspan="3">Location (LOCAL or REMOTE) of the SQL Database.
Default value=LOCAL</td></tr>
<tr><td>CALLTYPE</td><td><b>Target</b></td><td><b>GUI</b></td><td><b>TUI</b></td></tr>
<tr><td></td><td>CICS</td><td>LINK*
CALL</td><td>LINK*
CALL</td></tr>
<tr><td></td><td>DPS7</td><td>CALL*</td><td>CALL*</td></tr>
<tr><td></td><td>DPS8</td><td>LINK*
CALL</td><td>LINK*
CALL</td></tr>
<tr><td></td><td>IMS</td><td>CALL*</td><td>CALL*
CHNG</td></tr>
<tr><td></td><td>MICROFOCUS</td><td>CALL*</td><td>CALL*</td></tr>
<tr><td></td><td>TANDEM PATHWAY</td><td>PATHSEND</td><td></td></tr>
<tr><td></td><td>TUXEDO
Default value = TPCALL
value <b>not</b> to be entered</td><td>CALL</td><td>CALL</td></tr>
<tr><td></td><td colspan="3">* : default value</td></tr>
<tr><td>SERVBUFF</td><td colspan="3">This option identifies the Data Structure which defines the Server Buffer
SERVBUFF=xx
xx corresponds to the Data Structure code (2 character-code).
NB : the same option must be positioned at the level of the Folder Business Components Dialogue.
For more details on the Server Buffer, see section Options.</td></tr>
<tr><td>WAITINT</td><td colspan="3">Option specific to MQ Series and TCP-IP Socket on MVS/CICS
Waiting time, in seconds, between the starting of two queries
Default value for MQ Series = 0001
Default value for TCP-IP Socket on MVS/CICS = 1800
This default value can be modified by entering another four-character number</td></tr>
<tr><td>WAITINT1</td><td colspan="3">Option specific to MQ Series
Waiting time, in seconds, between the starting of the first request.
(default value = unlimited).
This default value can be modified by entering another four-character number</td></tr>
<tr><td>WORKFILB</td><td colspan="3"><b>Required option</b>
Database Block code for a work file in organization H or external name for a work file in organization V.
External name :
IBM CICS - VSAM : DDNAME (1 to 8 positions).
DPS8 - TP8 :
- If blocking by characters : external name on 4 characters (generated in the SELECT clause) and blocking on the last four characters (generated in the FD clause).
- If blocking by record : external name on 5 characters and blocking on the last three characters.
- the two parts of the external name, each on 2 characters, are separated by a dash.
For other variants : Logical name , generated in the SELECT clause.</td></tr>
<tr><td>WORKFILS</td><td colspan="3"><b>Required option</b>
Code of Segment associated with the work file <b>[*]</b>.</td></tr>
<tr><td>WORKFILO</td><td colspan="3"><b>Required option</b>
Organization of the work file used by the associated Monitor and Services Manager
V : sequential indexed
H : relational
W : working</td></tr>
</table>

[*] For technical reasons, you must create a Segment which defines a work file (for information on this work file's purpose, see the *Developer's Documentation / eBusiness and Pacbench/CS Applications: **Concepts & Architectures***). This Segment must belong to a Data Structure with any code and type. This Segment must call two Data Elements:

- a key Data Element, of type **U** and length 37,
- a Data Element with a minimum length of 250 and a maximum length equal to the message length minus 37 (the key length).

This Segment must be described in a Library higher than or equal to the Library where the Communications Monitor is described.

Example of the **Generation Parameters** *window of a Communications Monitor*



```
GENERATION OPTIONS FOR SCREE   CLCFOL

A LIN : T DESCRIPTION                                                LIB
  100 : O WORKFILO=V WORKFILS=FR00 WORKFILB=UT-S-FR                   2580
*** END ***                                                      C 1
```

☞ To generate a Communications Monitor, see Chapter 7: Generation and Extraction.

Implementing the Communications Monitor

For the Communications Monitor to be implemented, set the **LOCATION** option in the **Generation Parameters** window of the Folder Dialogue, or of the Business Components Dialogue if no Folder is specified. The value for this option (to be left adjusted) is the Communications Monitor code (6 characters) followed by a dash and the name of the environment (20 characters maximum).

Example:
LOCATION=FMCODE-ENVIRNAME

If you are using more than one ***LOCATION****, define each one of them – left aligned – on a separate line.*

Chapter 6: Folder and Folder Views (graphic applications)

A ***Folder*** occurrence is used to describe a set of elementary data aggregates and the functional relations which link them to create a complex information concept complete with access and processing services.

The generation of a Folder occurrence produces a **Services Manager** capable of interpreting and formatting all of the services associated with the Folder before calling the relevant Business Component.

A business act must be able to work with a partial view of a Folder, which guarantees the transformation of its instances; this is the ***Folder View***.

A Folder View is therefore always associated with a Folder.

A Folder can be represented by several Folder Views.

The coherence of a Folder View description in relation to that of its Folder is handled by the VisualAge Pacbase WorkStation.

The generation of a Folder View creates a **Folder View Proxy**, a set of classes which allow the Folder View to be managed in a graphic application.

Specifying Folder Views is not a requisite. Indeed, the description domain of a Folder may be limited enough to be used in full in a given application. In this case, an additional generation at the Folder level (**GVC** command) allows to obtain the whole set of classes used to manage the Folder in a graphic application.

The principles at work in a development that uses Folders are documented in the *Developer's Documentation / eBusiness and Pacbench/CS Applications:* ***Concepts & Architectures.***

Implementation Rules

Business Components and Folder Views must be defined and described in a Library higher than or equal to the Library where the associated Folder is defined. The Communications Monitor(s) (set in the Folder Dialogue's **LOCATION** option) follow the same rule. We will call this definition Library the "application Library".

If the application is to be run in different environments, you will need to use sub-Libraries of the application Library, called "conversion Libraries".

Each conversion Library corresponds to a specific generation variant.

Each Communications Monitor(s) is referenced at the Folder Dialogue level (**LOCATION** option) in the application Library, a Monitor being specific to an execution environment.

In addition, you will need more than one Monitor for a given conversion Library if more than one communication protocol is used under this environment (e.g., CICS ECI and CICS CPIC).

Business Components, Folders and Communications Monitors must be generated from their respective conversion Libraries.

However, Folder or Folder Views (GVC command) are always extracted from the application Library.

This structure calls for the following rules:

- You must not change a Folder's options or Descriptions (with the exception of the **LOCATION** option) and those of Folder Views in conversion Libraries. Also some fields in their Definitions must not be changed (example: lock option). These changes would not be taken into account when extracting Proxy objects from the application Library, this would result in serious inconsistencies.
- You must not change any Business Components in the conversion Libraries, except – this is necessary – for its generation variant.
- Of course, it is forbidden to change Definitions and Descriptions of Data Elements and Logical Views in these conversion Libraries.

Folder

Dialogue Level

The Folder Dialogue is a logical envelop which groups the Information System's Folders.

Important recommendations concerning architecture are given in *Subchapter Implementation Rules*.

The Folder Dialogue lets you specify general characteristics and generation variants and generation options which are used by default for all attached Folders.

Definition of a Folder Dialogue

To define a Dialogue is to create an occurrence of the **Dialogue** entity (2 character- **code**).

The data that you enter at the Dialogue level is used as default data at the Folder level.

The required fields at the Dialogue level are the following:

- the **name** of the Folder Dialogue
- the **type** **F** for Folder

⚠ Once entered, this value cannot be modified.

Setting Generation Options

A number of options can be used when generating a Folder. Since it is advised to enter some of these options in the Dialogue, they are documented in this Subchapter.

You enter these options on **O**-type lines, at the top of the **Generation Parameters** screen, as follows: **OPTION=option value** (with no space before and after **=**).

Several options, separated by at least a blank, may be entered on the same line.

No check is executed when you enter the options, but if they do not correspond to standard options, messages will be displayed in the generation report.

Other options are specific to each Folder. It is then important to enter them at the Folder level. For more details, refer to section Options.

<table>
<tr><td rowspan="10">CALLTYPE</td><td>Target</td><td>GUI</td><td>TUI</td></tr>
<tr><td>CICS</td><td>LINK*
CALL</td><td>LINK*
CALL</td></tr>
<tr><td>DPS7</td><td>CALL*</td><td>CALL*</td></tr>
<tr><td>DPS8</td><td>LINK*
CALL</td><td>LINK*
CALL</td></tr>
<tr><td>IMS</td><td>CALL*</td><td>CALL*
CHNG</td></tr>
<tr><td>MICROFOCUS</td><td>CALL*</td><td>CALL*</td></tr>
<tr><td>TANDEM PATHWAY</td><td>PATHSEND</td><td></td></tr>
<tr><td>TUXEDO
NB : default value =
TPCALL
Value **not** to be entered</td><td>CALL</td><td>CALL</td></tr>
<tr><td colspan="3">* : default value</td></tr>
<tr></tr>
<tr><td>CHANGE</td><td colspan="3">Option specific to TUXEDO :
Default value: Call of the Business Component with the TPNOCHANGE parameter of the CALL
If CHANGE=YES: Call of the Business Component with the TPCHANGE parameter of the CALL
NB: It is strongly recommended to set the same option in the Generation Parameters of the Business Components' Dialogue.</td></tr>
<tr><td>ERRLAB</td><td colspan="3">This option specifies how the errors must be returned.
By default, ERRLAB=YES : the Communications Monitor retrieves the key, gravity and error messages returned by the Business Component before sending back the answer to the Client Component.
But if the option value is NO, only the error key will be returned. In that case, the error message is locally read by the client application.</td></tr>
<tr><td>LOCATION</td><td colspan="3">Option required
VisualAge Pacbase code of the Communications Monitor (on 6 characters) followed by a slash and by the environment name (on 20 characters maximum), left adjusted.
Example :
LOCATION=FMCODE-NOMENVIR
If you use more than one LOCATION, enter each of them on a separate line, all left adjusted.</td></tr>
<tr><td>NULLMNGT</td><td colspan="3">Default value:
NULLMNGT=NO: No management of the presence of the Logical Views' Data Elements at the Proxy level
NULLMNGT=YES: Management of the presence of the Logical Views' Data Elements at the Proxy level
NB: If NULLMNGT=YES, the VECTPRES=YES option (in the Business Component Dialog) is systematically implemented.</td></tr>
<tr><td>TRAN</td><td colspan="3">Option specific to TUXEDO
Default value: TRAN=YES: Use of the transactional mode for the TUXEDO variant.
TRAN=NO: No use of the transactional mode
NB: It is strongly recommended to code the same option in the Generation Parameters of the Business Components Dialogue.</td></tr>
</table>

Specifying a Folder

Important recommendations concerning architecture are given in *Subchapter Implementation Rules*.

Definition of a Folder

To define a Folder is to create an occurrence of the **Folder** entity (6 character-**code**, the first two being identical to those of the Folder's Dialogue).

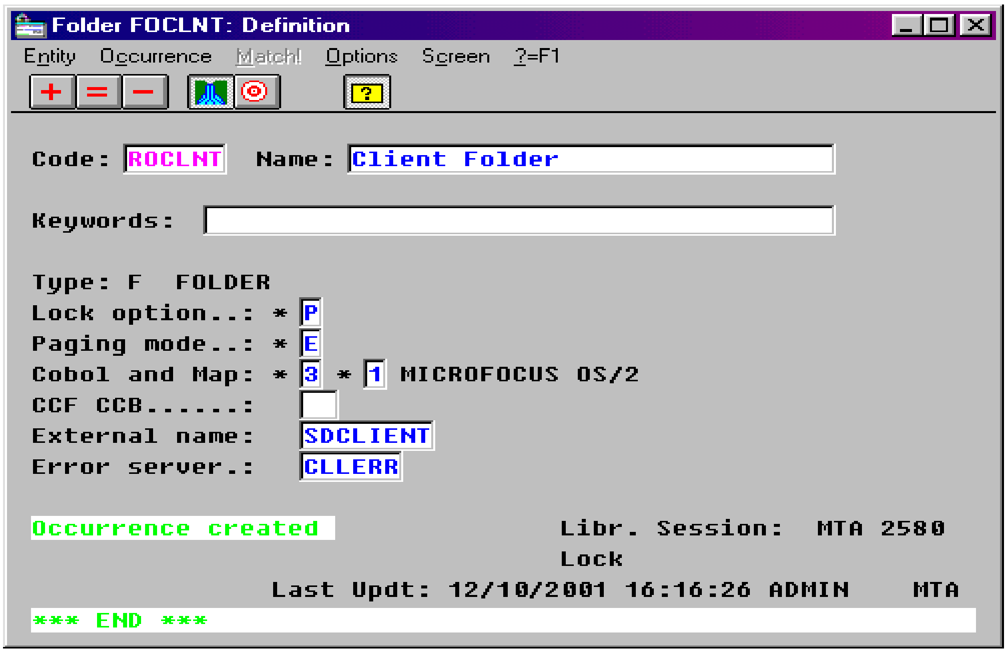
In the `Definition` window, you enter:

- the name of the Folder (required)
- one or more keywords (optional)
- the lock option (see below)
- the paging mode (see below)
- the generation variants
- the commands in front/in back
- the external name (default = Folder code)

  This value is used for the extraction of the Proxy via the GVC command.
- the Error Message Server code (required. Access to the error message server by clicking the right mouse button). This server is documented in *Subchapter* The Error Message Server.

If you press `F1` while the cursor is in an input field, on-line help is displayed; it contains a list of all possible values. If you double-click on the desired value, it will be automatically entered in the field.

Example of the `Definition` *of a Folder:*



```
Code: ROCLNT   Name: Client Folder

Keywords:

Type: F   FOLDER
Lock option..: * P
Paging mode..: * E
Cobol and Map: * 3 * 1 MICROFOCUS OS/2
CCF CCB.......:
External name:   SDCLIENT
Error server.:   CLLERR

Occurrence created               Libr. Session:   MTA 2580
                                 Lock
                  Last Updt: 12/10/2001 16:16:26 ADMIN     MTA
*** END ***
```

Lock Option

The *upload–download* mechanisms used in the management of Folders increase the time taken between the reading of the initial image of a Folder and the return of its updated image.

In this context, a lock process may adequately prevent simultaneous updates on the same Folder occurrence. In this manner, the coherence of accumulated updates is ensured.

You have a choice between two locking options: pessimistic mode and optimistic mode.

- The pessimistic lock mode (value **P** in the **LOCK OPTION** field) creates an exclusive appropriation of the Folder which is locked to all other users during updating.
- The optimistic lock mode (value **O**) is triggered only when the update is requested and compares the initially loaded Folder image with its current image in the database. If they are identical, the update is accepted.

☞ The setting of this option (values **P** or **O**) requires that locking and unlocking processing be written in the Business Component associated with the Folder's *root* node. For more information about inserting specific code, refer to section Lock / Unlock Service.

By default, no lock process is activated (**N** value).

Paging Mode

You can choose between two paging modes:

- *Non-extend* mode (value **N** in the **PAGING MODE** field) allows to page forward and backward through a predefined population. Each paging operation (forward or backward) executes a read request and its result replaces that of the previous read.
- *Extend* mode (value **E**) allows you to page forward through a predefined population, and the result of each paging adds to the preceding read. The reading of the previous pages is then handled by the graphic client.

By default, this option is activated in *non-extend* mode (**N** value).

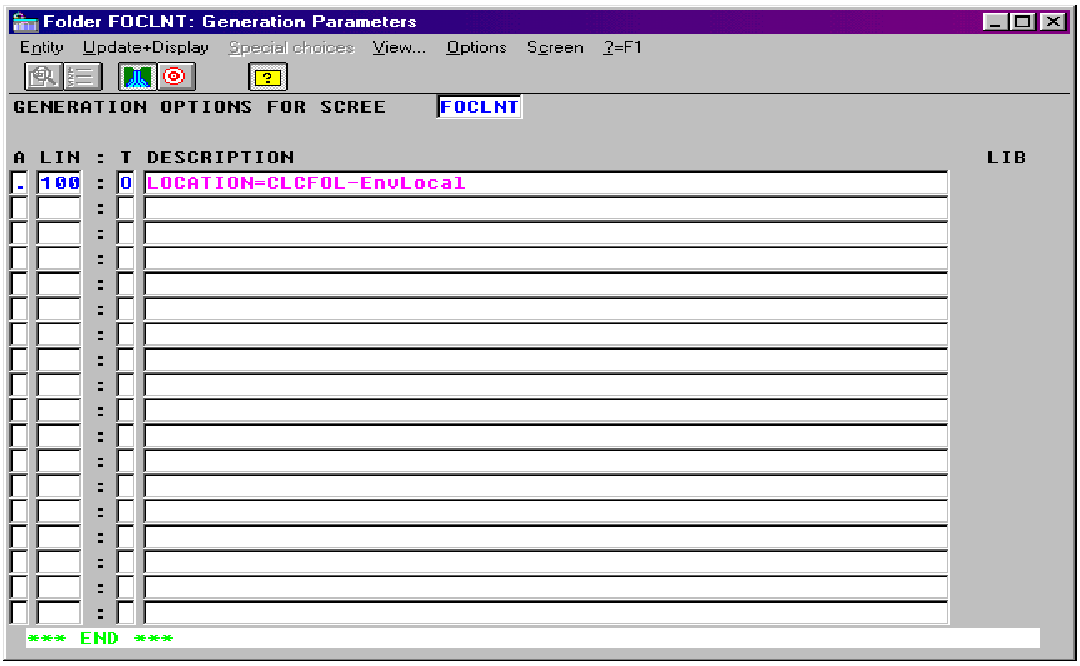
Options

The default generation options are those of the Folder *Dialogue* and are documented in section Setting Generation Options.

Options must however be set in the **Generation Parameters** window (**O**-type line) for each Folder.

INITSERV	**INITSERV =** VisualAge Pacbase code of the Initialization/Termination Business Component (I/T Component) This option must be left-aligned. This functionality is documented in *Subchapter* **Initialization/Termination Business Component**.
PROCESS	Available for TANDEM PATHWAY only. Name of the Process when calling the Folder. 15 characters maximum without space. There is no lowercase/uppercase conversion.

Example of the `Generation parameters` *window of a Folder:*



```
GENERATION OPTIONS FOR SCREE     FOCLNT

A LIN : T DESCRIPTION                                   LIB
. 100 : O LOCATION=CLCFOL-EnvLocal
*** END ***
```

Description of a Folder

The hierarchy of a Folder, or the tree structure of its nodes, is described in the `Business Components (semi local)` window.

The description of a Folder consists in creating its hierarchical node structure.

A Folder node corresponds to a Logical View managed by a Business Component.

There are three types of nodes:

- The ***root*** node (type `R`), unique in a Folder, is the parent of all the depending nodes. This node does not depend on any other node.
- The ***depending*** node (type `D`) is linked by a hierarchical relationship to one and *only one* root node or depending node.

  A root node or depending node can have one *or more* depending nodes.

  The cardinalities possible on this type of relation are:

  `(0,1), (0,n), (1,1), (1,n)`

  ☞ For large reading request of the Folder, for *each depending node*, the Segment(s) called by the Business Component attached to this node must bear the corresponding marker (for more information, see *Selection break* in section How to Access External Resources, paragraph *Use Characteristics*.
- The ***reference*** node (type `L`) – unlike the depending node – is linked by a referencing relation to the root node and/or to one *or more* depending nodes.

  A root or depending node can have one or *more* reference nodes.

  The cardinalities possible on this type of relation are:

  `(0,1), (1,1)`

Functional Impacts

An instance created in a depending node must depend on an instance of its root node or depending node.

The service that creates instances associated with a root node or a depending node is disabled when at least one of its depending nodes, linked by a minimum cardinality of 1, has not been defined in the corresponding Folder instance.

The deletion of an instance of a root node or depending node deletes all the instances of its depending nodes.

The deletion of an instance of a root node or depending node does not delete the instance of its reference node.

Updates of instances of reference nodes are forbidden.

Call order for Business Components:

Business Components are executed according to their call order in the `Business Components (semi local)` window, that is from left to right, and from top to bottom.

Description Rules

- A node code must be unique in the Folder description.

- Each set [Business Component / Logical View] which defines a root node or depending node must be unique in the sub-network (connection Library and higher-level Libraries), in other words, a root node or depending node can belong to one and only one Folder.

- All the Business Components and Logical Views called in a Folder description must be defined in the sub-network (connection Library and higher-level Libraries).

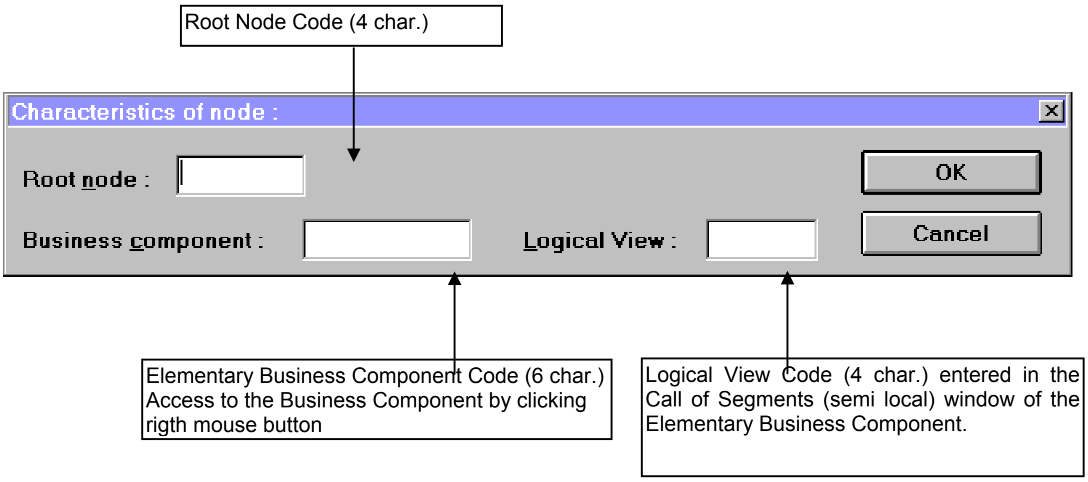
- In order to ensure the consistency of generation options, it is recommended that all the Business Components associated with the root and depending nodes of a Folder be attached to the same Dialogue.

How to Describe a Folder

The hierarchy of a Folder, or its tree structure in nodes, is described in the `Business Components (semi local)` window.

- To create a root node (type `R`):

Click on the only node displayed which initializes the tree structure (MASS node). Select `Insert node` in the `Edit` menu or press `INS`. A dialog box is displayed where you enter the characteristics of the root node that you wish to create:

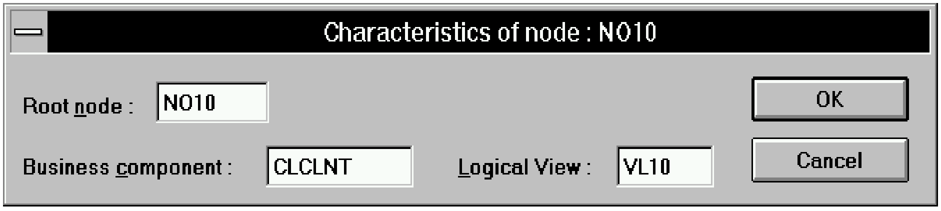


Root Node Code (4 char.)

Elementary Business Component Code (6 char.) Access to the Business Component by clicking rigth mouse button

Logical View Code (4 char.) entered in the Call of Segments (semi local) window of the Elementary Business Component.

*Example: Root node characteristics (*NO10*)*



Characteristics of node : NO10

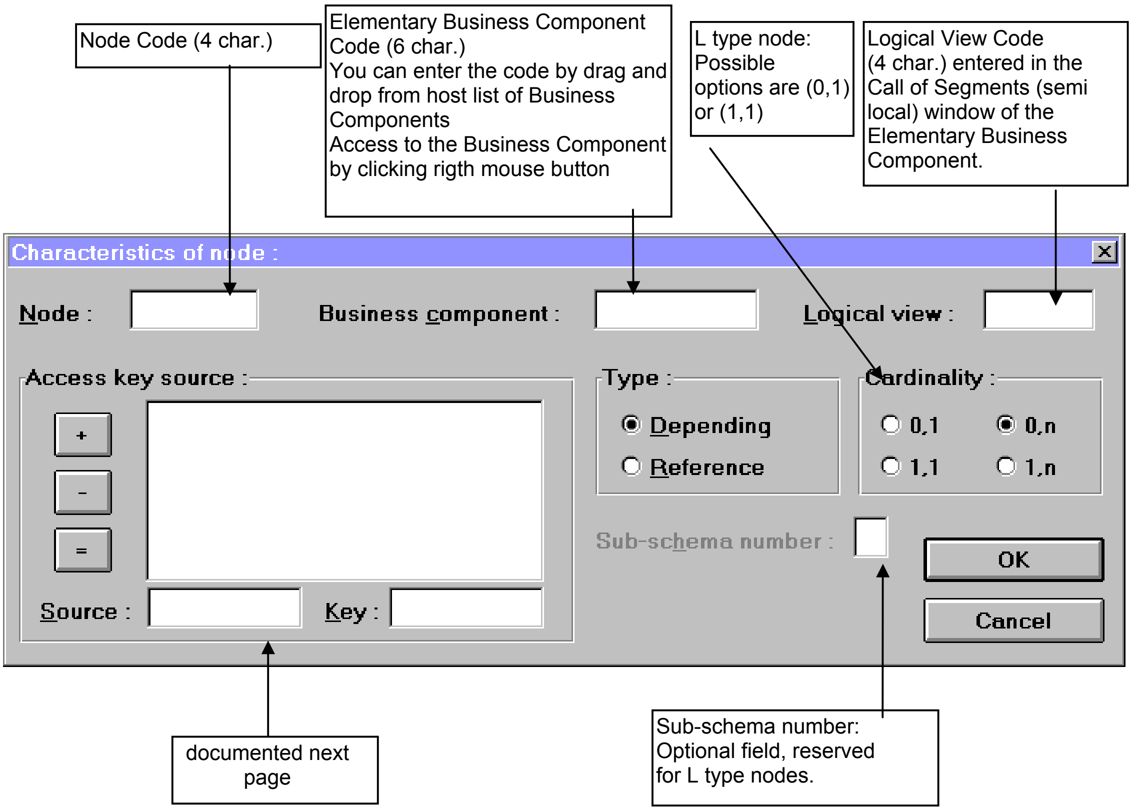
Root node : NO10

Business component : CLCLNT

Logical View : VL10

- To create a depending node (type **D**) or reference node (type **L**):

Click on the root node (or on a depending node that has already been created). Select **Insert node** in the **Edit** menu or press **INS**. A dialog box is displayed where you enter the characteristics of the node to create:



Node Code (4 char.)

Elementary Business Component Code (6 char.) You can enter the code by drag and drop from host list of Business Components Access to the Business Component by clicking rigth mouse button

L type node: Possible options are (0,1) or (1,1)

Logical View Code (4 char.) entered in the Call of Segments (semi local) window of the Elementary Business Component.

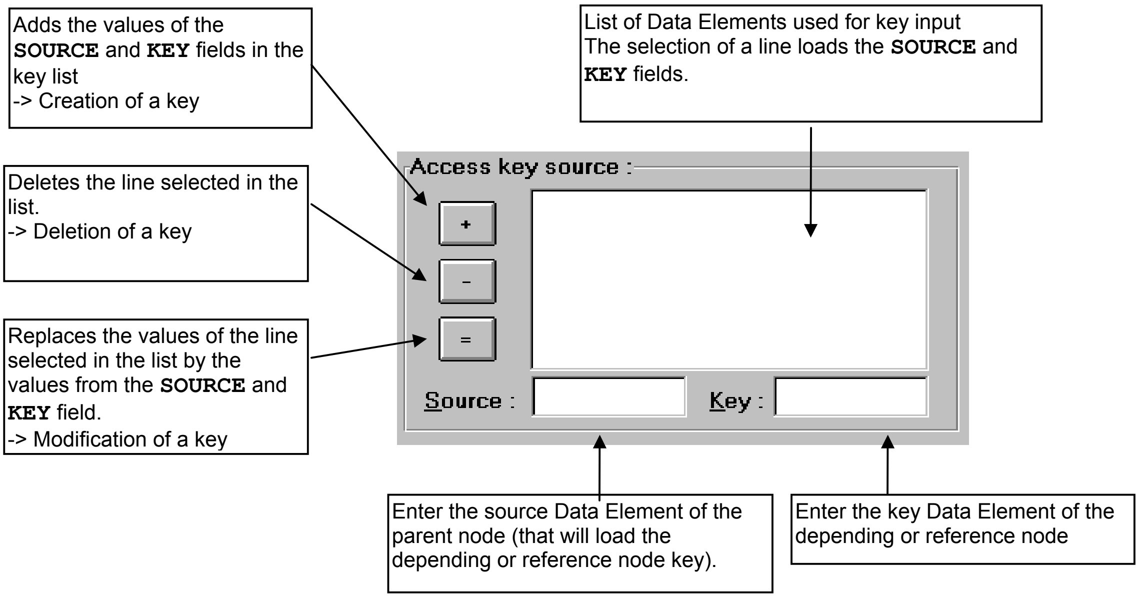
documented next page

Sub-schema number: Optional field, reserved for L type nodes.

- Access key source (for depending or reference nodes):

When a Data Element of the key corresponds to a Data Element in the parent node's key – identified by another code – you enter the code of the corresponding parent node's Data Element in the **Access key source** field.

☞ If a Data Element belonging to the key of a parent node also belongs to the key of a child node, it always corresponds. So it is impossible to specify an access key source for this Data Element.



Adds the values of the **SOURCE** and **KEY** fields in the key list
-> Creation of a key

Deletes the line selected in the list.
-> Deletion of a key

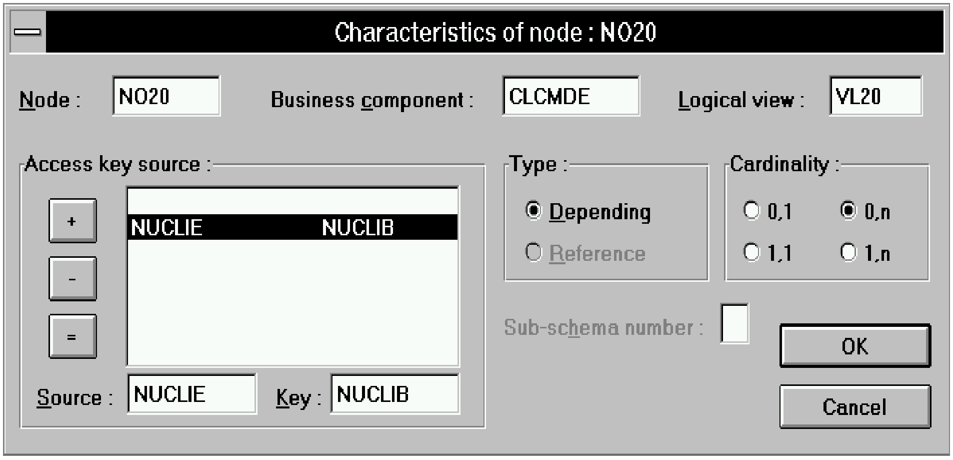
Replaces the values of the line selected in the list by the values from the **SOURCE** and **KEY** field.
-> Modification of a key

List of Data Elements used for key input
The selection of a line loads the **SOURCE** and **KEY** fields.

Enter the source Data Element of the parent node (that will load the depending or reference node key).

Enter the key Data Element of the depending or reference node

Example: Depending node characteristics (NO20)



Characteristics of node : NO20

Node : NO20 Business component : CLCMDE Logical view : VL20

Access key source :

NUCLIE	NUCLIB

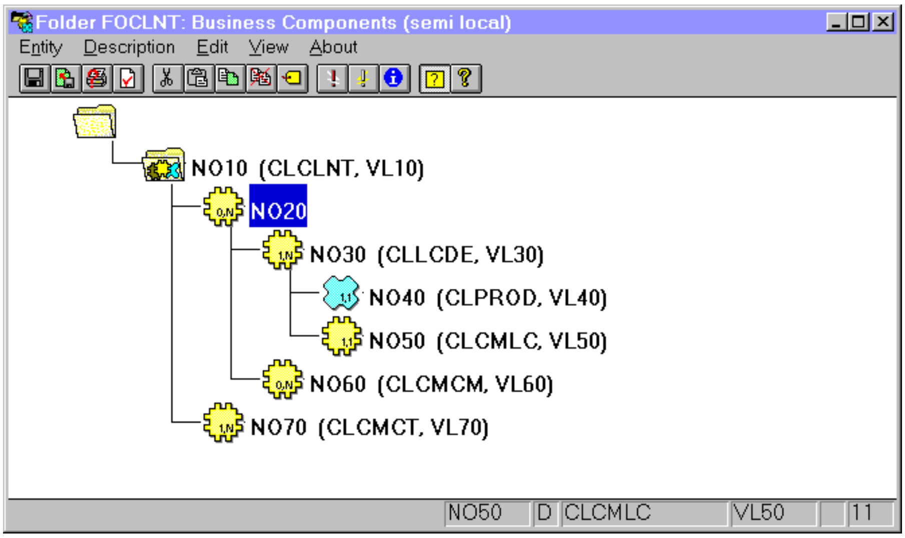
Source : NUCLIE Key : NUCLIB

Type : ● Depending ○ Reference

Cardinality : ○ 0,1 ● 0,n ○ 1,1 ○ 1,n

Sub-schema number :

Operations in the Business Components (semi local) Window



On the right of the node code, you find the code of the associated Business Component and Logical View.

You can access the Business Component associated with the node by clicking the right mouse button on the Business Component code.

- **Operations on the selected node (mouse click):**

You can change the selection in the tree structure by using the arrow keys (up or down).

- Adding nodes:
  - ♦ **`INS`** key or
  - ♦ Select **`Insert node`** in the **`Edit`** menu

  A dialog box is displayed where you can enter the characteristics of the node that you wish to create.

  It is forbidden to create a node under a reference node.

- Moving nodes:
  - ♦ left mouse button pressed or
  - ♦ select **`Cut/Paste`** in the **`Edit`** menu

  The node is moved along with *all* of its child nodes.

  It is forbidden to move a node below a reference node.

  When you move a node directly under the **`MASS`** node, it automatically becomes a root node (**`R`** type).

  When you move a node directly under the root node, it automatically becomes a depending node (type **`D`**).

- Deleting nodes:
  - ♦ **`DEL`** key or
  - ♦ select **`Delete node`** in the **`Edit`** menu

  The node is deleted along with *all* of its child nodes, after confirmation.

- Modifying nodes:
  - ♦ **`CTRL+ENTER`** or double-click, or
  - ♦ select **`Edit node`** in the **`Edit`** menu

A dialog box is displayed where you can edit the characteristics of the selected node.

Editing a node does not change its dependency links.

- Consulting nodes:
  - ♦ `ENTER` key or
  - ♦ select `Node Characteristics` in the `View` menu

  A dialog box is displayed showing the characteristics of the selected node.

  It is also possible to read the main characteristics of a node in the status bar, by simply placing the cursor over the corresponding line of the tree structure. You can "browse" through the main characteristics of each node in this manner.

- **Operations on the Folder Description:**
  - Checks before editing:

    When you open a non-empty Folder description, a series of checks is carried out locally. If any errors are detected, they are automatically displayed in a pop-up window, called `Download Errors`. You can also view these errors later on by reopening the window using the `F6` key or by selecting `Download Status` in the `View` menu.

  - Checks after editing:

    The validity of your Folder description is checked:

    - ♦ **Automatically**, during saving (select `Upload` in the `Description` menu or `CTRL+M`), and when the window is closed (by clicking, `ALT+F4` or selecting `Quit` in the `Description` menu).

      These checks are carried out in *two* steps:

      - . the *first* step makes local checks on the internal coherence of your Folder Description.
        If any errors are detected during this first step, they are automatically displayed in a pop-up window. The upload is stopped until all the errors are corrected.
        You can also view these local errors later on by reopening the window using the `F7` key or selecting `Upload Status` in the `View` menu.

        ☞ If in the meantime, you have modified the Folder and requested a validation (see below **On request**), the contents of this window will be modified accordingly.

      - . the *second* step makes checks on the coherence relative to the subnetwork of the VisualAge Pacbase Dictionary (connection Library and higher-level Libraries).
        If any errors are detected, they are automatically displayed in a pop-up window. The upload is completed when all the errors have been corrected.
        You can also view these errors later on by reopening the window using the `F7` key or selecting `Upload Status` in the `View` menu.

- **On request**, by selecting `Controls` in the `Description` menu or `CTRL+L`.

These checks – local only – verify the internal coherence of your Description.
If any errors are detected, they are automatically displayed in a pop-up window. You can also view these local errors later on by reopening the window using the F7 key or selecting Upload Status in the View menu.

Folder Views

Specifying Folder Views is not a requisite. Indeed, the description domain of a Folder may be limited enough to be used in full in a given application. In this case, an additional generation at the Folder level allows to obtain directly the whole set of classes used to manage the Folder in a graphic application.

Dialogue Level

The Folder View Dialogue is a logical envelop which groups Folder Views.

Important recommendations concerning architecture are given in *Subchapter Implementation Rules*.

The Folder View Dialogue lets you specify general characteristics and generation variants and options which are used by default for all attached Folder Views.

The **Dialogue entity** allows to create and describe the Folder View Dialogue of the application.

Definition of a Folder View Dialogue

To define a Folder View Dialogue is to create an occurrence of the Dialogue entity (2-character-**code**).

The data that you enter at the **Dialogue** level is used as default data at the Folder View level.

The required fields at the **Dialogue** level are the following:

In the `Definition` window, you enter:

- the **name** of the Folder View Dialogue
- the **type** for Folder View: `FV`

⚠ Once entered, this value cannot be modified.

Specifying a Folder View

Important recommendations concerning architecture are given in *Subchapter Implementation Rules*.

Definition of a Folder View

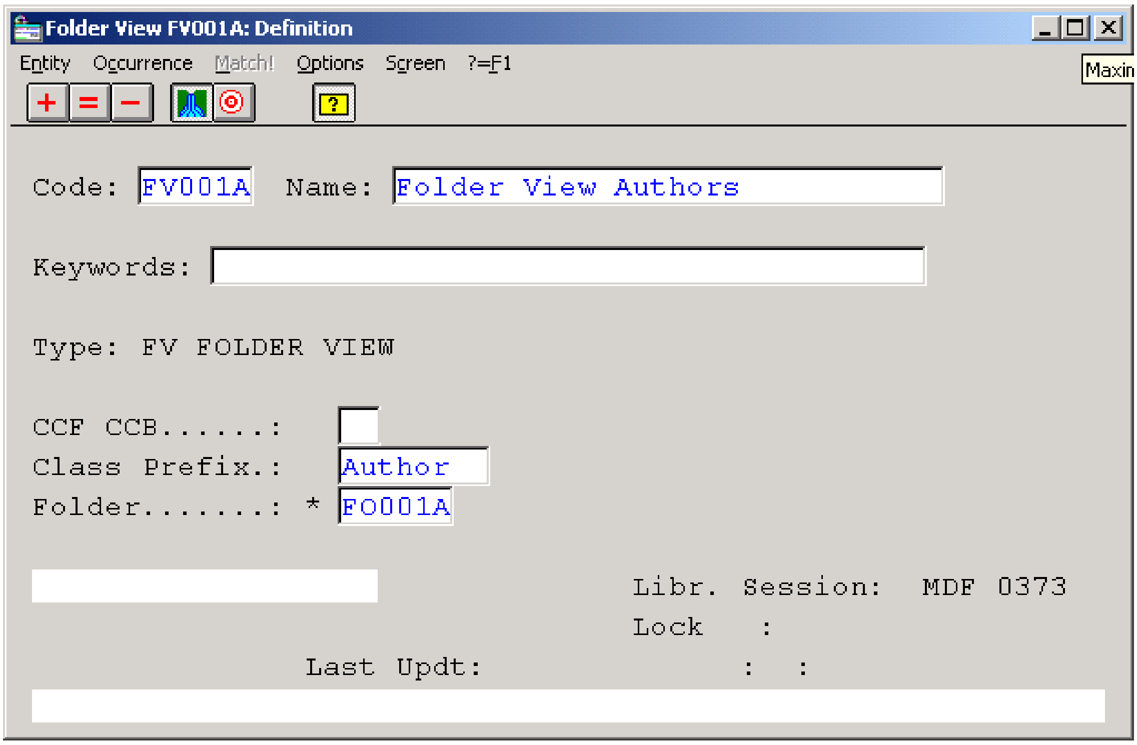
To define a Folder View is to create an occurrence of the Folder View entity (6 character-**code**, the first two being identical to those of the Folder View Dialogue).

In the `Definition` window, you enter:

- the **name** of the Folder View (required)
- one or more **keywords** (optional)
- the **commands in back/in front**
- the **Folder** to which the Folder View belongs (required)

If you press `F1` while the cursor is in an input field, on-line help is displayed; it contains a list of all possible values. If you double-click on the desired value, it will be automatically entered in the field.

Example of the `Definition` *of a Folder View:*



```
Code: FV001A   Name: Folder View Authors

Keywords:

Type: FV FOLDER VIEW

CCF CCB.......:
Class Prefix.:    Author
Folder.......: *  FO001A

                                   Libr. Session:  MDF 0373
                                   Lock    :
                   Last Updt:              :  :
```

Description of a Folder View

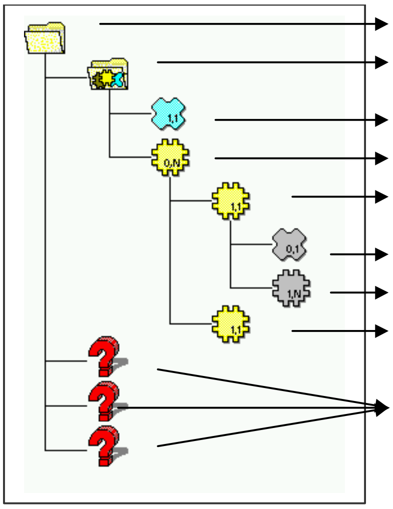
A Folder View is a partial representation – a sub-set – of the Folder it is attached to, and is described in the `Business Components (semi local)` window.

The first time the View is opened, all the Folder nodes are grayed; the View must be composed.

Manipulations in this window are limited to consultation and inclusion or exclusion in the Folder View of nodes existing in the Folder.

As a result, the choices in the `Edit` menu are different, but all other operations described in section How to Describe a Folder are available for a Folder View.

The different graphical representations of a Folder View's nodes:



Tree structure root, included in the Folder View

Root node (type R), included in the Folder View

Reference node (type L), included in the Folder View

Depending node (type D), included in the Folder View

Depending node, included in the Folder View

Reference node, not included in the Folder View

Depending node, not included in the Folder View

Depending node, included in the Folder View

3 incoherent nodes

Including Nodes in the Folder View

A node must be selected in the View with its parent nodes, all the way to the root node.

Two selection modes are available:

- A selection from the selected node to the root node, including all parent nodes:

  Press **`CTRL`** and click on the node or select **`Include with Parent Nodes`** in the **`Edit`** menu.

- A global selection which includes the entire branch of the selected node:

  Press **`SHIFT`** and click on a node of the branch or select **`Include the Whole Branch`** in the **`Edit`** menu.

  The nodes included in the Folder View are no longer grayed.

Excluding Nodes from the Folder View

A node is excluded from a View with its depending nodes all the way to the last node in the branch.

Two deletion modes are available:

- An exclusion from the selected node to the last node, including all depending nodes:

  Press **`CTRL`** and click on the node or select **`Exclude with Child Nodes`** in the **`Edit`** menu.

- A global exclusion which includes the entire branch of the selected node:

  Press **`SHIFT`** and click on a node of the branch or select **`Exclude the Whole Branch`** in the **`Edit`** menu.

The nodes that you have excluded from the Folder View are grayed out.

Checks on the Folder View

- Checks before editing:

  When you open a non-empty Folder View description, a series of checks is carried out locally. If any errors are detected, they are automatically displayed in a pop-up window, called **`Download Errors`**. You can view these errors later by reopening the window using the **`F6`** key or by selecting **`Download Status`** in the **`View`** menu.

- Checks after editing:

  The validity of the Folder View is checked:

  - ♦ **Automatically**, during saving (select **`Upload`** in the **`Description`** menu or **`CTRL+M`**), and when the window is closed (by clicking, **`ALT+F4`** or selecting **`Quit`** in the **`Description`** menu).
    These checks are carried out in *two* steps:
    - . the *first* step makes local checks on the internal coherence of your Folder View.
      If any errors are detected during this first step, they are automatically displayed in a pop-up window. The upload is stopped until all the errors are corrected.
      You can also view these local errors later on by reopening the window using the **`F7`** key or selecting **`Upload Status`** in the **`View`** menu.

      ☞ If in the meantime, you have modified the Folder View and requested a validation (see below), the contents of this window will be modified accordingly.
    - . the *second* step makes checks on the coherence relative to the subnetwork of the VisualAge Pacbase Dictionary (connection Library and higher-level Libraries).
      If any errors are detected, they are automatically displayed in a pop-up window. The upload is completed when all the errors have been corrected.
      You can also view these errors later on by reopening the window using the **`F7`** key or selecting **`Upload Status`** in the **`View`** menu.
  - ♦ **On request**, by selecting **`Controls`** in the **`Description`** menu or CTRL+L. These checks – local only – verify the internal coherence of your Description.
    If any errors are detected, they are automatically displayed in a pop-up window. You can also view these local errors later on by reopening the window using the **`F7`** key or selecting **`Upload Status`** in the **`View`** menu.

Managing Inconsistencies in the Folder

A node is said to be inconsistent when a discrepancy appears regarding this node between the Folder View and its Folder.

- **Deleting a node in the Folder**:

When an existing Folder View is displayed, a node which no longer belongs to the Folder is placed under the MASS node with a specific icon [question mark].

Make sure that this is not an error in the Folder, then delete this node:

Press **DEL** and click on the node or select **Exclude Node** in the **Edit** menu.

- **Adding a node to the Folder**:

When an existing Folder View is displayed, a node which has been added to the Folder is shown grayed out.

- If this node is located between two nodes already included in the Folder View, make sure that this is not an error in the Folder. It must then be included in the Folder View:

  Press **INS** and click on the node or select **Include Node** in the **Edit** menu.

- If this node is not located between two nodes already included in the Folder View, you should reconsider the composition of the View taking into account this new node.

Chapter 7: Generation and Extraction

Generation

Business Components, Services Managers, Communications Monitors, and Error Message Servers are generated by the **GPRT** Generation-Print batch procedure.

Use the **GGS** command followed by the occurrence code. You can use more than one **GGS** command in a **GPRT** execution.

☞ For more details on the **GPRT** procedure, refer to the *VisualAge Pacbase* Operations Manual, *The Developer's Procedures*.

These generations must be executed from the conversion Library of the occurrences concerned.

The result of the generation is contained in a file whose name depends on the VisualAge Pacbase execution platform. To find out the name of this file, refer to the Operations Manual described above.

Particular cases

Generating a Business Component in single-view mode

The positioning of the **LOCATION** option triggers the generation of two COBOL sources:

- The first one corresponds to the generation of the Business Component selected,
- The second one corresponds to the Services Manager source (it is a purely technical component which is used to generate all the facilities generated by the Folder).

In the Business Component **Definition,** you can inhibit the generation of the:

- Business Component with the value **$** entered in the **CCF** field (Commands in front), on the **Program** line.
- Services Manager with the value **$** entered in the **CCF** field (Commands in front), on the **Complement** line.

Generating a Communications Monitor TCP-IP Socket under MVS/CICS

At the **LINK** stage, it is necessary to implement a specific library which take into account the API Socket services:

```
// SYSLIB  DD DSN=PT$VIC.TCPIP310.SEZATCP,DISP=SHR
INCLUDE SYSLIB(EZACICAL)
INCLUDE SYSLIB(EZACIC04)
INCLUDE SYSLIB(EZACIC05)
```

Proxy Extraction (graphic applications)

The extraction which is done before the generation of a Proxy is executed by the **GPRT** procedure.

For more details on the GPRT procedure, refer to the *VisualAge Pacbase* Operations Manual, *the Developer's Procedures*.

To extract a **Folder View**, use the GVC command followed by the occurrence code of the Folder View.

To extract a **Folder**, use the GVC command followed by the occurrence code of the Folder. All the Folder nodes are taken into account by the GPRT procedure.

The extraction of Folder Views is always executed from the application Library.

- **Single-view development**:

To extract a **Logical View**, connect to the application library, use the GVC command followed by the occurrence code of the **Business Component**.

You can use more than one GVC command in a GPRT execution.

GGS and GVC commands are incompatible within one GPRT execution.

The result of this extraction is contained in an 80-character sequential file, whose name depends on the VisualAge Pacbase execution platform.

To find out the name of this file, refer to the Operations Manual mentioned above.

When you have created the extraction file, the FVP (Folder View Proxy) must be generated. If you have used a single-view development, the Proxy resulting from the Logical View extraction is, once imported in the Client component, considered as a FVP for its programming interface is that of a FVP.

The generation, importation and development of the graphic Client are documented in the *Pacbench C/S User's Guide, Volume III – Graphic Clients*.

Chapter 8: Appendix : Summary of the Generated Business Component

☞ The generated Business Component is not presented here in full. Only the sections which contain useful data for the development of an application are shown here.

Working Storage Section

Beginning of Working Storage Section

Level WSS-BEGIN

The Working Storage Section begins with the WSS-BEGIN level. It contains all the variables and keys necessary for automatic processing.

IK Return code for a Segment access or a Business Component call:

0 No error

1 Error

CATM Transaction code:

C Creation

M Modification

A Deletion

X Implicit update

ICATR Indicator of the current repetition (repeated Logical View's data)

OPERS Operation code

IRR Number of repetitions requested by the client

OPERV Logical View processing indicator

V Logical View recognized by the Business Component

blank Logical View not processed by the Business Component

OPER2 Operation code for the call of a second-level Business Component

IER Maximum number of errors on database accesses that the Business Component can detect before returning to the Client. This number is determined by the ACCESERR option.

IED Maximum number of database access errors that the Business Component is able to detect before returning to the Client. This number is determined by the DATAERR option.

ICATRC Number of requested repetitions during the call of a Business Component.

DEL-ER Memorizes Data Element error (work variable)

OPERB Indicates if the selection requested is a list at a Business Component all

OPERT Area for the management of check/update/selection services

Level **PACBASE-CONSTANTS**

The **PACBASE-CONSTANTS** level gathers generation context information:

SESSI Session number of the generation Library

LIBRA Library code

DATGN Program generation date (format based on the language code)

Example: *DD/MM/YY if user language =’F’*
MM/DD/YY if else

PROGR Library Program code

PROGE Program external name

TIMGN Program generation time

USERCO User code

COBASE Database code

DATGNC Program generation date with century (format based on the language code)

Example: *DD/MM/CCYY if user language =’F’*
MM/DD/CCYY if else

Level **V-INFO-CODE**

The **V-INFO-CODE** level also gathers generation context information i.e.:

Code of the Logical View

Code of the Pactables table
Access to a Pactables table : code of the Pactables table.

File or table external name
Access to an indexed file or a SQL table: File or table external name.

Extract name
Access by extract method: extraction method name.

SEGMENT CODE
Non-standard customized accesses: Segment code.

ROLLBACK

VWER

Code of the Logical View used in the Error Server.

TPABORT	(specific to TUXEDO)
TPBEGIN	(specific to TUXEDO)
TPCALL	(specific to TUXEDO)
TPCOMMIT	(specific to TUXEDO)

This list is not complete. Other values may be used for a number of accesses. These are specific to a given hardware and vary according to their use in the Folder.

Description of Segments

For each Logical View Segment (**VIEW** in our example), two `02` levels are generated, redefining each other.

`02 view-C` Data transferred in the direction Client --> Server and in both directions.

`02 view-S` Data transferred in the direction Server --> Client.

When the Logical View contains repeated data, a `01` level is also generated:

`01 RE-view-delco`

Repeated data (view=view code, delco=Group Data Element for repeated data)

`10 1-view-delco`

Identifier description of the Logical View's repeated structure. This Data Element is declared only once in the Logical View, and its type is **U**.

Additional Descriptions

Work Indicators

K50L Work indicator of the number of Segment access errors.

K50D Work indicator of the number of Data Element errors.

Data Element Errors

Errors are stored in Working-Storage Section, in areas prefixed with **EE-VIEW**.

These areas are generated for each Logical View **VIEW** called in the Business Component. This table allows to store the errors detected on the Logical View using the **ERR** operator. The number of items in the table depends on the **DATAERR** option. Each item contains the information needed to format the key used to access the error message file.

EE-view-LIBRA	Library Code
EE-view-SERVER	Business Component code
EE-view-VIEW	Logical View code
EE-view-DATCOD	Data Element code
EE-view-DATERR	Error code
EE-view-DATTYP	Error type
S	Standard error
U	User error
EE-view-ICATR	Number of the repetitive line on which the error was detected

Description Buffer

A-CATM Description buffer. This buffer contains the transaction codes related to both non repeated and repeated data of the Logical View.

A-CATM-CA Action code applying to non repeated data. Systematically generated.

A-CATM-CR OCCURS N

Action code applying to repeated data.

N is the maximum number of repetitions for all the Logical Views in this server. Generated if one of the Business Component's Logical View contains repeated data.

Segment Access Errors

Errors on Segment are stored in Working-Storage Section, in areas prefixed with **V-ERR**.

V-ERR This is a non-specialized area which is always generated. It is used to store database access errors. The **OCCURS N** depends on the **ACCESERR** option.

V-ERR-BUFFER OCCURS N

V-ERR-SEGCOD	Erroneous Segment code
V-ERR-SEGERR	Error code
V-ERR-SEGTYP	Error type

V-ERR-ICATR rank of the erroneous occurrence in a multi-occurrence processing

V-ERR-LIBRA Library code

V-ERR-SERVER Business Component code

Check and Presence of Data Elements

CH-view view represents the Logical View.
This area contains a table used for the checks of the Logical View. It is generated if the CHECKSER=YES option has been specified in the **Generation Parameters** screen of the Dialogue (or Business Component) This table contains one item for each Data Element called in the Logical View. The possible values (to be specified by the user in the Client component) are:

N No check on this Data Element (default value)

P Missing Data Element

blank All Data Elements are checked

This area is also used to indicate if the value of a field in the Logical View is null or not (in SQL standard), i.e. if it is present. For example, the Client can set null values for an update service and the Business Component can return null values for the selection (if the corresponding fields in the table are null).

Closing of the Cursor

L-CURS-EXTNAM

The closing of the cursor is conditioned at the end of a selection service if an extraction method is used.

Linkage Section

The LINKAGE SECTION, i.e the communication area, is executed when the Client is called (or in the Client Monitor according to the used architecture).

It is executed in three steps:

- Loading of the communication area,
- Call of the Business Component with running of the communication area,
- Retrieval of the message at the Business Component return.

⚠ The following fields **MUST NOT** be modified:

TECH-BUFFER Technological buffer

TECH-LGCOMM Global length of the communication area run by the Client Component.

TECH-LGTECH Length of the TECH-BUFFER area, always equal to 320.

TECH-LGUSER Length of the User Buffer. The buffer is optional.

TECH-LGVIEW Length of the Logical View to be processed (a Business Component can process several Logical Views, one at a time, but the length of each of them can be different).

TECH-LGDESC Length of the description buffer.

TECH-LGERR Length of the buffer of errors on the Logical View's data.

TECH-STRUCT Not used, but must be set to s by the Client Component.

⚠ You could possibly – but it is left to your own responsibility – modify the following areas. As they are standardly specified, it is recommended to use intermediary areas described in the WORKING STORAGE SECTION to modify them. Caution: you must not add or modify existing values of these areas.

TECH-CLIENT Code of the calling Client. Not used for a Client calling Business Components in synchronous mode.

TECH-VIEW Code of the Logical View to be processed.

TECH-SERVER Code of the Business Component called.

TECH-OPER Service requested by the Client for the Logical View. The possible values are the following:

L monoinstance read service

A selection service (generally a BROWSE from one or more criteria)

E check service (no data is returned)

X check, selection service

M check, update service

T check, update, selection service

U user service

TECH-LIBRA Not used.

TECH-ICATRC Number of instances to be processed by the Business Component for the Logical View.

TECH-ICATRS Number of instances selected by the Business Component for a selection or check, update, selection service.

TECH-IERRS Indicator of non-system errors for selection access, set by the Business Component:

0 no error

L no error, end of list

S error

C instance not found

TECH-IERRU Indicator of non-system errors for check or update access, set by the Business Component:

0 no error

C access error

U the TECH-IERRU indicator is set to 'U' value in the following 3 cases :

- Data control from logical view,
- ERU operator,
- ERR operator.

TECH-IERRC Indicator of non-system set by the Business Component:

0 no error

A non-recoverable error on file or DBMS physical access

L length error for the communication area or the different buffers

O error on service request

S value error for **TECH-STRUCT**

V Logical View code not known by the Business Component

D different version number in the Client and the Business Component

TECH-IDATAC Check indicator of the Logical View's data, set by the Business Component:

blank check on the data of the Logical View whose check indicator contains a check request (see the description of the indicator in **CONT-BUFFER**).

N no check

C check of all the Data Elements

TECH-ERRCOD File status or SQLCODE in the case of physical access error. Set by the Business Component.

This area is split up into **TECH-STATUS** area

TECH-CODE Name of the file or the table in which a non-recoverable error has been detected. Set by the Business Component**.**

TECH-TYPE File access type in the case of a non-recoverable error, set by the Business Component (**R** for read**, W** for write, ...).

TECH-LGREAL Real length of the communication area set by the Client and the Business Component.

TECH-EXTNAM Extraction method code set by the Client.

TECH-TRANS Beginning/end of transaction indicator set by the Client:

B beginning of transaction

E end of transaction

TECH-COMMIT Indicator of the COMMIT requested by the Client:

- blank no COMMIT request
- C COMMIT request
- R ROLLBACK request

TECH-LGDATA Length of the check indicator of the Logical View's data set by the Client.

TECH-NUVERS Number of the version used for the compatibility check between the Business Component and the graphic client.

TECH-BROWSE Selection type indicator:

- blank direct selection by selection criterion
- B browse
- M direct selection by the Logical View's data

TECH-SRVUSR User Service initialized by the Client.

TECH-LUW

- blank TUI Client application
- S Server LUW application
- C Client LUW application

TECH-REQST Management of the call sequence of Business Components after a client query

- F First Business Component called
- M Intermediary Business Component called
- L Last Business Component called
- C Single Business Component called

TECH-CALSRV Management of the open and close sequence of cursors for large reading instance selection

- blank Default
- F First call of the Business Component to carry out selection
- M Intermediary call of the Business Component to carry out selection
- L Last call of the Business Component to carry out selection
- C Selection carried out using one call

TECH-TIMEST Timestamp management for a lock or unlock service

TECH-FILL Internal filler, reserved.

USER-BUFFER Optional User Buffer. This buffer is the same within a Dialogue.

SELT-BUFFER End-of-access buffer on a Segment in selection or repetitive category. This buffer contains the following fields:

SELT-SEGCOD Segment code

SELT-SEGERR Code of error on Segment

SELT-SEGTYP Type of error (S: standard) on Segment

SELT-LIBRA Library code of the Business Component which has detected the end-of-access on Segment

SELT-SERVER Code of the Business Component which has detected the end-of-access on Segment.

CONT-BUFFER When no error is detected, this buffer contains the application data, which must be conveyed as Logical Views between the Client and the Business Component. When an error is detected, this buffer is redefined by the **ERR-BUFFER** area, documented below.

ERR-BUFFER Error buffer, divided into errors on Segments and errors on Data Elements:

- **ERR-BUFSEG** Segment access error buffer:
- **ERR-SEGCOD** Code of the erroneous Segment
- **ERR-SEGERR** Error code:
  - **DUPL** Creation by mistake of a record, already existing record
  - **NFND** Modification or deletion by mistake, non-existing record
  - **END** End of list
  - **ABSC** Record to be selected not found
- **ERR-SEGTYP** Error type
- **ERR-ICATR** Line number of the erroneous occurrence for a multi-occurrence processing
- **ERR-LIBRA** Library code
- **ERR-SERVER** Business Component code
- **ERR-DATA** Buffer of errors on Data Element
- **ERR-LIBRA** Library code
- **ERR-SERVER** Business Component code
- **ERR-VIEW** Logical View code
- **ERR-DATCOD** Erroneous Data Element code
- **ERR-DATERR** Error code
- **ERR-DATTYP** Error type
- **ERR-ICATR** Line number of the erroneous occurrence for a multi-occurrence processing.
- **FILLER** End filler

Procedure Division

Each generated processing is identified by a tag.

```
05                   Begin Server                   FSERVER
05                   SQL Declaration                FSQL
               10    Whenever Clause                FSQL-WHENEVER
               10    Declare Cursor                 FSQL-CURSR-SEGT
05                   Logical View                   FVIEW
               10    Logical View init.             FVIEW-BEGV
               10    Check/update                   FVIEW-CHUP
                15   Multi-occurrence processing    FVIEW-CHUP-CATR
                 20  Check service                  FVIEW-CHUP-CATR-SRVE
                  25 Initializations                FVIEW-CHUP-CATR-SRVE-INIT
                  25 Checks                         FVIEW-CHUP-CATR-SRVE-CHCK
                  25 End of service                 FVIEW-CHUP-CATR-SRVE-DONE
                 20  Read service                   FVIEW-CHUP-CATR-SRVL
                  25 Initializations                FVIEW-CHUP-CATR-SRVL-INIT
                  25 Checks                         FVIEW-CHUP-CATR-SRVL-CHCK
                  25 Transfers                      FVIEW-CHUP-CATR-SRVL-TRAN
                  25 End of service                 FVIEW-CHUP-CATR-SRVL-DONE
                 20  Chck/updt service              FVIEW-CHUP-CATR-SRVM
                  25 Initializations                FVIEW-CHUP-CATR-SRVM-INIT
                  25 Checks                         FVIEW-CHUP-CATR-SRVM-CHCK
                  25 Transfers                      FVIEW-CHUP-CATR-SRVM-TRAN
                  25 Update                         FVIEW-CHUP-CATR-SRVM-UPDT
                  25 End of service                 FVIEW-CHUP-CATR-SRVM-DONE
                 20  Chck/updt/sel service          FVIEW-CHUP-CATR-SRVT
                  25 Initializations                FVIEW-CHUP-CATR-SRVT-INIT
                  25 Checks                         FVIEW-CHUP-CATR-SRVT-CHCK
                  25 Transfers                      FVIEW-CHUP-CATR-SRVT-TRAN
                  25 Update                         FVIEW-CHUP-CATR-SRVT-UPDT
                  25 End of service                 FVIEW-CHUP-CATR-SRVT-DONE
                 20  Chck/selection                 FVIEW-CHUP-CATR-SRVX
                  25 Initializations                FVIEW-CHUP-CATR-SRVX-INIT
                  25 Checks                         FVIEW-CHUP-CATR-SRVX-CHCK
                  25 End of service                 FVIEW-CHUP-CATR-SRVX-DONE
               10    Lock                           FVIEW-LOCK
               10    Unlock                         FVIEW-UNLK
               10    Selection                      FVIEW-SELC
                15   Multi-occurrence processing    FVIEW-SELC-CATR
                 20  Selection service              FVIEW-SELC-CATR-SRVA
                  25 Initializations                FVIEW-SELC-CATR-SRVA-INIT
                  25 Selection                      FVIEW-SELC-CATR-SRVA-SELC
                  25 Transfers                      FVIEW-SELC-CATR-SRVA-TRAN
                  25 End of service                 FVVIEW-SELC-CATR-SRVA-DONE
               10    User service                   FVIEW-USER
               10    Service errors                 FVIEW-ERR
               10    End of Logical View            FVIEW-END
05                   End of Server                  FSERVER-END
                     (return to the Client)
```

Elementary processing performed from the program body:

```
          10     Data transfers to Segment              FVIEW-TRDT
           15    Multi-occurrence transfers             FVIEW-TRDT-CATR
if CHECKSER=YES:
          10     Data check for the View                FVIEW-CHKD
           15    Multi-occurrence checks                FVIEW-CHKD-CATR
            20   Check on Element DELCO1                FVIEW-CHKD-CATR-DELCO1
            20   Check on Element DELCO2                FVIEW-CHKD-CATR-DELCO2
                 etc.
          10     Data transfers to View                 FVIEW-TRVW
           15    Multi-occurrence transfers             FVIEW-TRVW-CATR
          10     Logical access to Segment for check    FSEGT-CHCK
           15    Multi-occurrence access                FSEGT-CHCK-CATR
            20   Key loading                            FSEGT-CHCK-CATR-ALIM
            20   Call of physical access                FSEGT-CHCK-CATR-CALL
            20   Error handling                         FSEGT-CHCK-CATR-ERRS
          10     Logical access to Segment for updt     FSEGT-UPDT
           15    Multi-occurrence access                FSEGT-UPDT-CATR
            20   Key loading                            FSEGT-UPDT-CATR-ALIM
            20   Call of physical access                FSEGT-UPDT-CATR-CALL
            20   Error handling                         FSEGT-UPDT-CATR-ERRS
          10     Logical access to Segment for slct     FSEGT-SLCT
           15    Multi-occurrence access                FSEGT-SLCT-CATR
            20   Key loading                            FSEGT-SLCT-CATR-ALIM
            20   Call of physical access                FSEGT-SLCT-CATR-CALL
            20   Error handling                         FSEGT-SLCT-CATR-ERRS
           15    Access to one occurrence               FSEGT-SLCT-CATT
            20   Key loading                            FSEGT-SLCT-CATT-ALIM
            20   Call of physical access                FSEGT-SLCT-CATT-CALL
            20   Error handling                         FSEGT-SLCT-CATT-ERRS
```

Physical accesses performed from the logical accesses and generated in function 80:

```
 10     Read for check                         F80-SEGT-R
 10     Read for selection                     F80-SEGT-RA
 10     Read for check before update           F80-SEGT-RU
 10     Read of first record                   F80-SEGT-P
 10     Read of next record                    F80-SEGT-RN
 10     Creation (write)                       F80-SEGT-W
 10     Modification (rewrite)                 F80-SEGT-RW
 10     Deletion (delete)                      F80-SEGT-D
 10     Unlock of the record read in RU        F80-SEGT-UN
```

General processing performed and generated in function 81.

```
 10     Calculation of the real length of
        the communication area                         F81CA
 10     Processing of non-recoverable error            F81ER
 10     Close                                          F81FI
 10     Setting of processing to be performed
        in case of error (CICS)                        F81HC
 10     Call of TUXEDO routine for error
        handling                                       F81MS
 10     Return to the Client                           F81RE
 10     Date check                                     F8120
```

The function 81 also includes the retrieval of errors; the corresponding sub-functions vary according to the persistent data storage structure.

Examples:

```
10          Physical access error handling          F81-ES
            (VSAM files)
10          Commit (Oracle)                         F81-CM
10          SQL error                               F81-EV
```