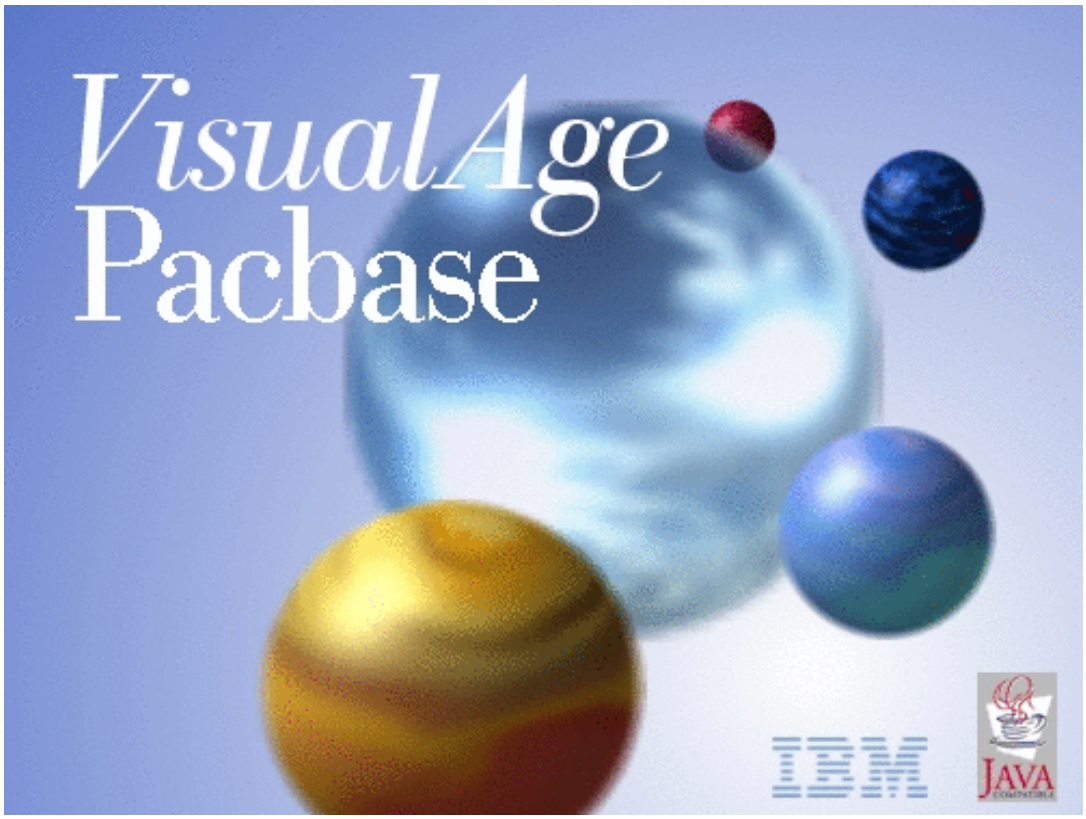
VisualAge Pacbase

API COBOL User's Guide

Version 3.5



VisualAge Pacbase

API COBOL User's Guide

Version 3.5

Note

Before using this document, read the general information under “Notices” on page v.

According to your licence 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 (March 2004)

This edition applies to the following licensed programs:
- 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
1, place Jean–Baptiste Clément
93881 Noisy-le-Grand, France.

Contents

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 the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk NY 10504–1785, U.S.A.

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 the information which has been exchanged, should contact IBM Paris Laboratory, SMC Department, 1 place J.B.Clément, 93881 Noisy-Le-Grand Cedex. 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.

Chapter 1. Introduction

The purpose of this manual is to give you all the required information to use the COBOL API via Developer workbench.

With the COBOL API, you can use Folders and Elementary Components within application programs and independently of the use of Proxies. In these programs, you can benefit from the Folder's information, hierarchy of information and collection of process. Thus, you can use the Folder and enter external eBusiness Application inputs.

Example: To manage automatic deductions from its customers' accounts at a fixed date, a bank can use a Program entity to operate the information of the 'customers' Folder.

You are advised to write the application program in VA Pac Program entity. However, it is also possible to write it with the Elementary Component or with the Screen entity.

The COBOL API allows you to specify service requests (upate, edition of lists...) on the Folder called in your application program. To write easily your program, you have at your disposal:

- structured language operators that will format the message in input to the Folder.
- the COBOL description of the data in the communication area in input to the Folder.

To generate the application program including the eBusiness entities call, you need to run the GPRC batch procedure. For more details on this procedure, refer to the 'Developer's guide' for your platform, in chapter 'Generation and printing'.

Chapter 2. Specifying the work areas

Introduction

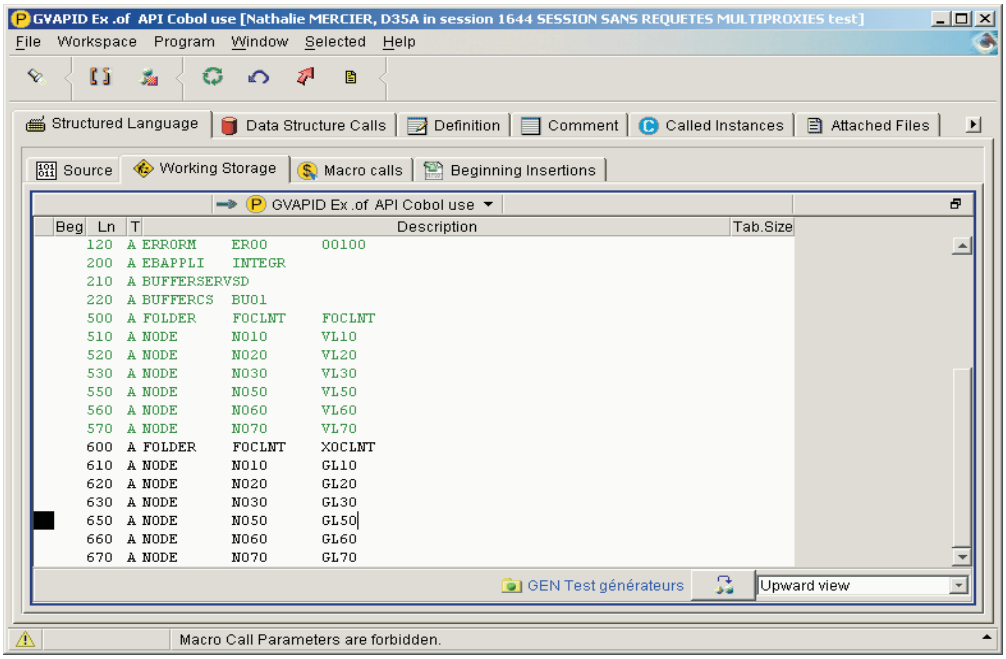
The first step consists in defining the eBusiness Application and its Folders in the work areas of your application program. You must then:

- specify which operator will be responsible for managing error messages returned by the services,
- call the eBusiness Application of your wish,
- indicate which buffers are used by the Application, if any,
- call the Folders and the Folder's nodes used.

This information must be indicated in the 'Structured Language' tab of your application program, in the 'working storage' sub-tab, on the lines of 'A' type.

The 'Description' section of these lines is divided into fields made up of 10 characters. You must indicate the code of the API COBOL operator in the first field, and the parameters of the operator in the other fields (the number of parameters depends on the operator specified)

Caution: to be able to use the COBOL API properly, you must be respectful of the contents and format of the values you enter in the fields as there is no input control yet in this current release.



Error management

You must first specify which operator is used to manage the errors returned by the services executed.

```
A ERRORM    PARAM1    PARAM2

Where

 PARAM1=XXXX  code of the Segment generated
 PARAM2=nnnnn number of items for this structure
              (nnnnn=00001)
```

All the structures required to manage errors will be generated in the work area of the application program where you specified the operator.

The code defined by PARAM1 is the program Segment code of the structure which identifies the code and the error codes. It is compulsory and must be different of the code of the existing structure of the program.

In the generated COBOL, the structure defining errors returned by the services is the following one:

```
01                XXXX.
   05             XXXX-ERRNB     PICTURE 9(5) VALUE ZERO.
   05             XXXX-WERR.
     10           XXXX-ERR          OCCURS nnnnn.
      15          XXXX-ERRKEY    PICTURE X(25).
      15          XXXX-LABEL     PICTURE X(67).
   05             XXXX-ERRCOD.
     10           XXXX-STATUS    PICTURE X(09).
     10           XXXX-CODE      PICTURE X(27).
     10           XXXX-TYPE      PICTURE X(02).
```

ERRNB is the number of errors sent back by the Folder manager after the execution of a service for the eBusiness Application.

ERR is including these errors, identifiers and messages.

ERRCOD includes SQL errors.

CODE includes the 'SQL ERRCODE' value .

TYPE and STATUS include error codes.

Specifying an eBusiness application

To specify an eBusiness Application in your application program, do as follows:

```
A EBAPPLI   PARAM1

Where

PARAM1=XXXXXX Repository code (short code) of the
              eBusiness Application.
```

Linkings will be built on the eBusiness Application.

You can call only one eBusiness Application per application program.

All the structures required to specify the Application will be generated in the work areas of the application program where the instruction is specified.

Specifying a user buffer

If the eBusiness Application is using a user buffer, you must specify the buffer as follows:

```
A BUFFERCSPARAM1
Where
 PARAM1 = XXXX  Code of the Segment generated
```

This Segment code is set to the repository code of the Segment defining the user buffer of the eBusiness Application. You can modify its name in the application program.

In the gererated COBOL, the complete structure of the user buffer in the eBusiness Application, is defined as follows:

```
01 XXXX.
  10 XXXX-CORUB1.
   15 XXXX-CORUB2 PIC X(nnn).
  10 etc.
```

Specifying a server buffer

If the eBusiness Application is using a server buffer, you need to define the buffer as follows:

```
A BUFFERSERVPARAM1
Where
PARAM1 = XX Code of the Structure generated
```

The Data Structure code is set to the repository code of the Data Structure defining the Data Structures of the server buffer in the eBusiness Application. You can change its name in the application program.

This instruction is used to generate the structures of the server buffer in the eBusiness Application under the form of a FILLER, as the buffer is used only in the Elementary Components.

```
01 XX00.
  05 FILLER  PICTURE X(nnnnn).
```

Specifying Folders

To declare Folders in the application program you need to specify for each Folder, which Folder and nodes are used.

A Folder must be declared as follows:

```
A FOLDER    PARAM1    PARAM2
Where  PARAM1 = XXXXXX  Repository code (short code) of
                        a Folder in the eBusiness Application.
    PARAM2 = XXXXXX  Generated code of the Folder
```

Linkings are built from the Folder's repository code.

PARAM2 is set to the repository code. You can modify its name in the application program.

One Folder can be defined several times in the program until you give them different names.

This instruction is used to generate all the work areas used in the different Folder's service requests.

Specifying Folder's nodes

Once the Folders used are declared, you need to specify which nodes are used by your application program.

To declare a Folder's node, do as follows:

```
A NODE      PARAM1    PARAM2
```

```
Where
 PARAM1= XXXX Node's identifier in the corresponding Folder
 PARAM2= XXXX Program code of the node
```

The program code of the node is set to the node's identifier in the Folder. You can modify the name of a node in the application program.

You can have different occurences for one node but make sure that they are defined with different program codes.

The generated COBOL includes:

- the structure of the Logical View's identifiers,
- the structure of the Logical View associated with the node,
- the list of extraction methods associated with the Logical View,
- the description of a Logical View occurence and the corresponding presence indicator,
- the list of user services defined for the node.

STRUCTURE OF THE LOGICAL VIEW IDENTIFIERS

Caution: this information is given only to be consulted.

```
01       ID-xxxx-LEN    PIC 9(5) VALUE 00009.
```

Indicates the length of identifiers.

First level of backup for the identifiers of the selection:

```
01        1-ID-xxxx.
     10     ID-xxxx-ident1 PICTURE nnnn
     10     ...
```

Second level of backup for the identifiers of the selection:

```
01        2-ID-xxxx.
     10     ID-xxxx-corub1 PICTURE nnnn
     10    ...
```

STRUCTURE OF THE LOGICAL VIEW ASSOCIATED WITH THE NODE

This structure includes the action codes, the presence indicator, the identifiers, the extraction parameters and the Logical View instances. This part corresponds to the structure which is found in input of the Elementary Component.

```
01             xxxx.
  02         A-xxxx.                              Action code
    05       A-xxxx-CATM.
       10    A-xxxx-CA     PICTURE X.
       10    A-xxx-CR      PICTURE X OCCURS nnnn.
```

```
  02       CH-xxxx.                      Presence indicator
   05      CH-xxxx-xxxx
               OCCURS nnnn.
       10  FILLER        PICTURE X(nnnnn).
  02       ID-xxxx.                              Identifiers
       10  1-xxxx-ident1 PICTURE xxxx
       10  ..
  02       EP-xxxx.                 Extraction parameters
       10  2-xxxx-param1 PICTURE nnnn
  02       xxxx-xxxx       OCCURS nnnn.   Log. View inst.
       15  FILLER        PICTURE X(nnnnn).
```

In case of update services, you are responsible for loading the first instance of the action codes, the presence indicator and the fields of the View.

In case of selection services, you need to load the identifiers and extraction parameters.

LIST OF EXTRACTION METHODS ASSOCIATED WITH THE LOGICAL VIEW

```
01         EM-xxxx.
    05     EM-xxxx-EXTNAM   PIC X(10).
    05     EM-xxxx-CODES.
      10   FILLER  PIC X(10) VALUE "xxxxxxxxx ".
      10   ..
    05     EM-xxxx-CODESR REDEFINES EM-xxxx-CODES.
      10   EM-xxxx-VCODES  PIC X(10) OCCURS nnnnn.
```

In case of a selection with the use of extraction methods, the EXTNAM field must contain the code of the method used.

DESCRIPTION OF A LOGICAL VIEW INSTANCE AND OF THE CORRESPONDING PRESENCE INDICATOR

```
01         RE-xxxx-xxxx.
        15  xxxx-corub1  PICTURE xxxxxx
      15  ...
01         RH-xxxx-xxx.
    05      CH-xxxx-NUCLIE PICTURE X.
    05      CH-xxxx-NOMCLI PICTURE X.
```

LIST OF USER SERVICES

```
01         US-xxxx.
     05    US-xxxx-SRVUSR PIC X(25).
     05    US-xxxx-CODES.
       10  FILLER  PIC X(25) VALUE "PRINT".
       10  ..
     05    US-xxxx-CODESR REDEFINES US-xxxx-CODES.
       10  US-xxxx-VCODES   PIC X(25) OCCURS nnnn.
```

When using a user service, the SRVUSR field must include the code of the service.

Chapter 3. Service request on a Folder

Introduction

With the COBOL API, you can specify in your application program, service requests on the Folder. To do so, specify the instructions which are used to manage:

- updates,
- readings or collections,
- user services,
- initialization and terminaison services.

You have to specify an instruction in the 'Structured Language' tab of your application program, in the 'Source' sub-tab, , with EXS as operator.

The 'Operand' part of the line on which you enter the EXS operator is divided into fields made up of 10 characters. You need to indicate in the first operand field, and then in the following ones, the parameters needed for the use of the operand (the number of parameters depends on the operand used).

Caution: to be able to use the COBOL API properly, you must be respectful of the contents and format of the values you enter in the fields as there is no input control in this current release yet.

Update service

To indicate an update service, proceed as follows:

```
EXS UPDATE    PARAM1    PARAM2    PARAM3    PARAM4
Where
 PARAM1  = XXXXXX      Program code of the Folder
 PARAM2  = XXXX        Program code of the node
 PARAM3  = 0 or 1      Control option
 PARAM4  = 0 or 1      Refresh option
```

To be able to work correctly, the eBusiness application and the Folder must be defined in the work areas.

You must initialize by yourself the structures required to execute the service properly, that is to say, the presence indicators of the Logical View structure, the action code and possibly the user buffer.

You must test whether the service is running correctly. To do so, check that the number of errors (ERRNB field in the program generated) is not superior to 0 at the service return.

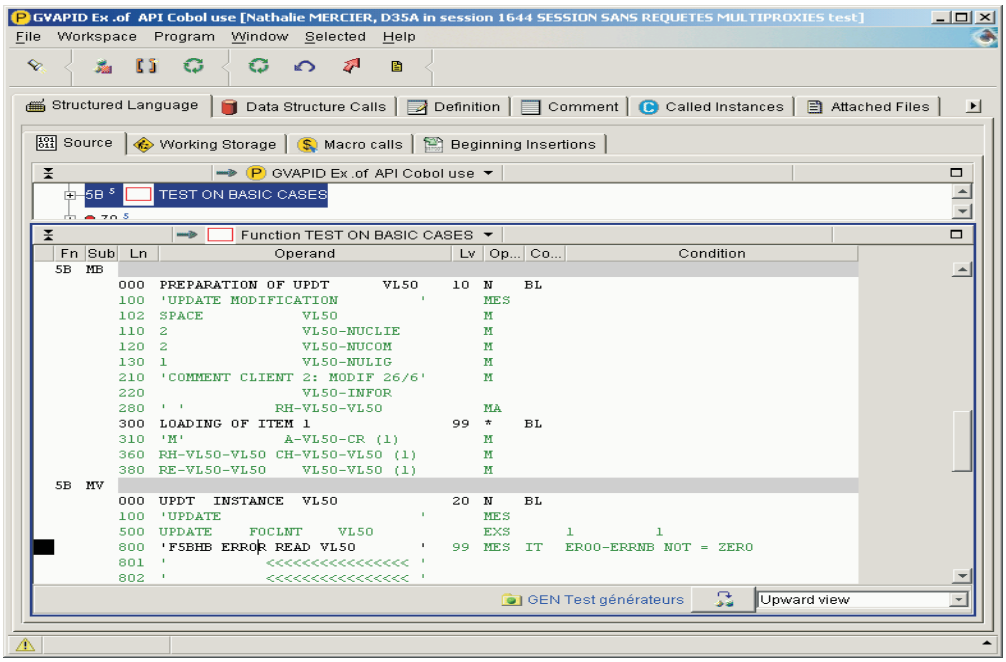
In case of an update, you must perform a consistency control of the Folder (cardinality control for example) and specify the commit and rollback instructions.

If the refresh option is specified, the presence indicators and the structure of the Logical View associated with the node are initialized at the service return.

Each time a creation order is sent to a dependent node, the Folder Manager controls that the mother instance exists. If it does not exist, an error is returned

and the instance which was suggested is not created. Therefore, you must be respectful of the nodes hierarchy in the Folder.

In case of a delete, the service systematically delete the instance and all the dependent ones if the 'Delete Cascade' option was selected in the Elementary Component.



LOCK/UNLOCK OF A FOLDER

```
EXS LOCK      PARAM1    PARAM2
EXS UNLOCK    PARAM1    PARAM2
```

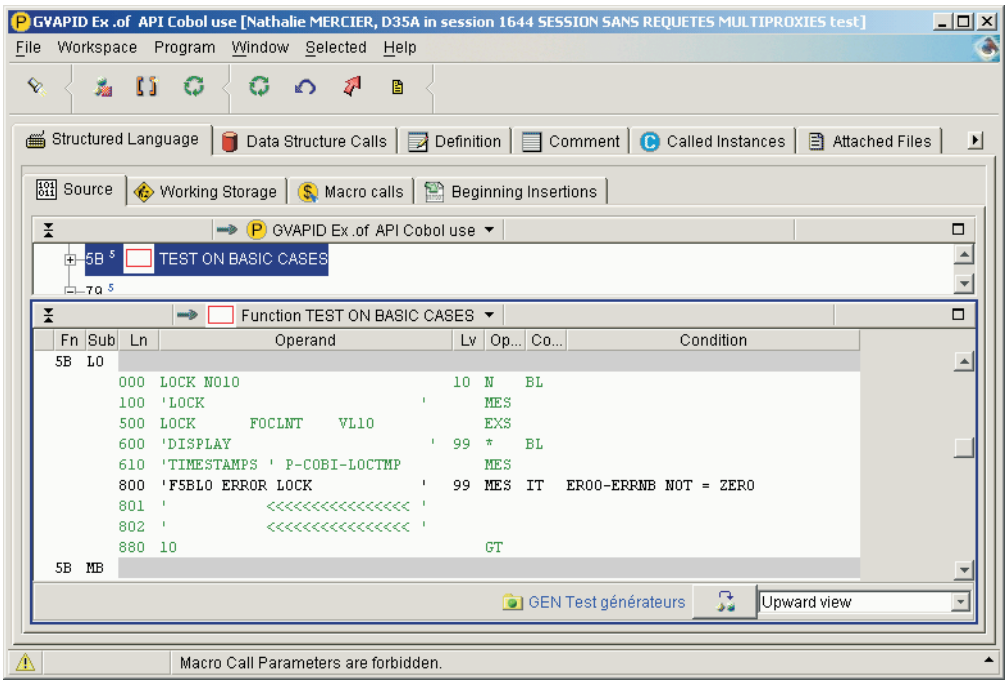
Where

```
 PARAM1  = XXXXXX Program code of the Folder
 PARAM2  = XXXX   Program code of the hierarchical node
```

To run correctly, the eBusiness Application and the Folder must have been defined in the work areas.

You must initialize the structures required to run the service properly, that is to say, initialize the structure of the Logical View's identifiers.

You also need to test whether the service runs correctly. To do so, check that the number of errors (ERRNB field in the generated program) is not superior to 0 at the service return.



Read service

Two types of read services do exist:

- services whose result corresponds to one instance at the most. Only one operator is required to specify this type of services.
- services whose result can be superior to one instance. Two operators are required for this type of services: the first one is used to define the properties of the appropriate collection while the second one is used for a sequential read of the collection from the beginning to the end.

All the read services can be executed outside an update step.

Reading with at the most one instance as a result

To indicate a read service sending back only one instance, do as follows:

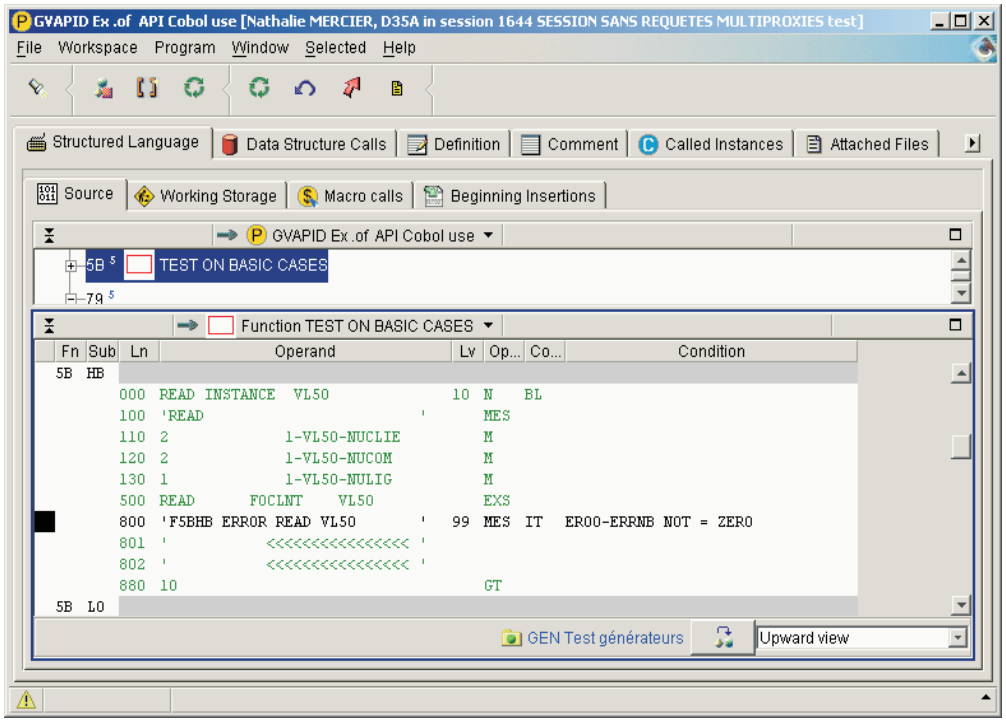
```
EXS READ      PARAM1    PARAM2
Where:
 PARAM1 = Program code of the Folder
 PARAM2 = Program code of the node
```

This operator reads a Folder's node instance from its key, and if the instance exists, sends it back to the structure of the associated Logical View with its presence indicators. When an error is met or if the instance does not exist, an error message is positioned and the ERRNB field is incremented. Whatever the result may be, the user buffer is initialized.

The eBusiness Application and Folder must have been defined in the work areas to be able to run correctly.

You must initialize the structures required to ensure the smooth running of the service, that is to say, the structure of the Logical View identifiers and if necessary, the structure of the user buffer.

To test the service execution, check that the number of errors (ERRNB field in the generated program) is not superior to 0 at the service return.



Reading with at least one instance as a result

For this service, you need to specify two operators:

- the first one is used to define the collection
- the second one is used to read the collection in a loop or a succession of performs.

The definition of a collection includes the parameters required to the read process of one or several nodes.

Collection definition

The first step consists in defining the collection as follows:

```
EXS OPENCOLLECPARAM1    PARAM2    PARAM3    PARAM4    PARAM5

Where

  PARAM1 = XXXXXX  Program code of the collection
  PARAM2 = XXXXXX  Program code of the Folder
  PARAM3 = XXXX    Program code of the node

  PARAM4 =         Selection option
           S  Single node (default value)
           F  First children
           A  All children

  PARAM5 =         Option for the selection result
           H  Hierarchical
           N  By Node (default value)
```

PARAM1 must be unique in the application program and is to be used to the naming of private fields required to manage the collection.

It is possible to open several collections on the same Folder and node.

This first operator either prepares the reading of a collection of instances on a Folder's node from a start key, or reads a Folder's node and all the instances associated with the first level child nodes or associated with all its child nodes. The start key required always corresponds to that defined in the structure associated with the node defined by PARAM3. Each node concerned in the collection can recognize an extraction method which is defined in the EM-param3-EXTNAM field of the structure associated with the node. The extraction parameters must be initialized in the corresponding structure.

You must initialize the structures required to ensure the smooth running of the service.

The 'hierarchical' option allows to retrieve the collection instances in the hierarchical order. The 'byNode' option retrieves the instances node by node.

The service doesn't recognize the contents of 'resultOptionList' when the 'singleNode' option is positioned.

The eBusiness Application and Folder should have been defined in the work areas to be able to run correctly.

To test whether the service is running correctly, check that the number of errors (ERRNB field in the generated program) is not superior to 0 at the service return.

Collection tracing

Now, you must specify the second operator which will allow you to read the collection so as to download the instances selected.

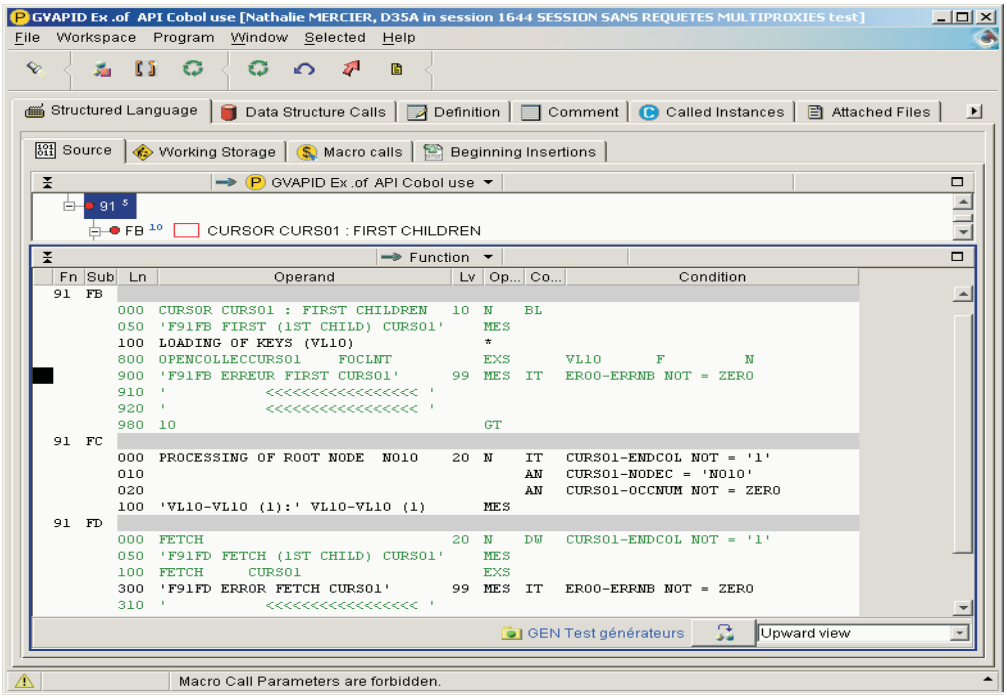
```
EXS FETCH     PARAM1

Where
  PARAM1  = XXXXXX Use code of the collection
```

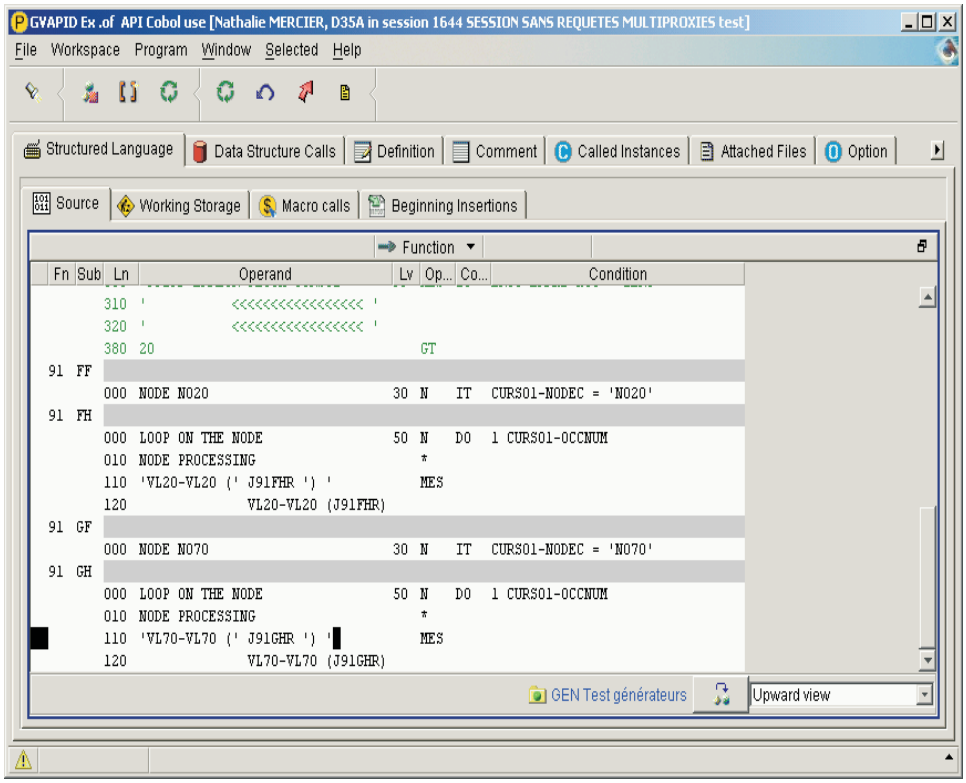
This operator sends back a node instance of the collection identified by PARAM1. This instance is transfered to the structure of the Logical View corresponding to the node as being executed.

Each time an instance is fetched, the service initializes the presence indicators of the instance too.

The end of list on a node is sent back to the structure receiving error messages at the same time as the last instance of the node.

To test whether the service is running correctly, check that the number of errors (ERRNB field in the generated program) is not superior to 0 at the service return.





WORK AREAS GENERATED

The work areas indicated below are generated to read a collection. They are available on consultation only; they should not be modified. These are technical data used by the Folder and its Elementary components.

ENDCOL is the end of collection indicator. It takes the '1' value when the end of the collection is reached, otherwise it takes another value.

ENDNOD is the end of node indicator. It takes the '1' value at the end of the node, otherwise it takes another value.

VIEWI represents the initial Logical View.

VIEWC represents the current Logical View.

NODEI represents the initial node.

NODEC represents the current node.

OCCNUM represents the number of instances returned to the Logical Views table.

NBREC represents the total number of records read for this collection.

User service

To specify a user service request, do as follows:

```
EXS EXUS      PARAM1    PARAM2
Where
 PARAM1 = Program code of the Folder
 PARAM2 = Program code of the node
```

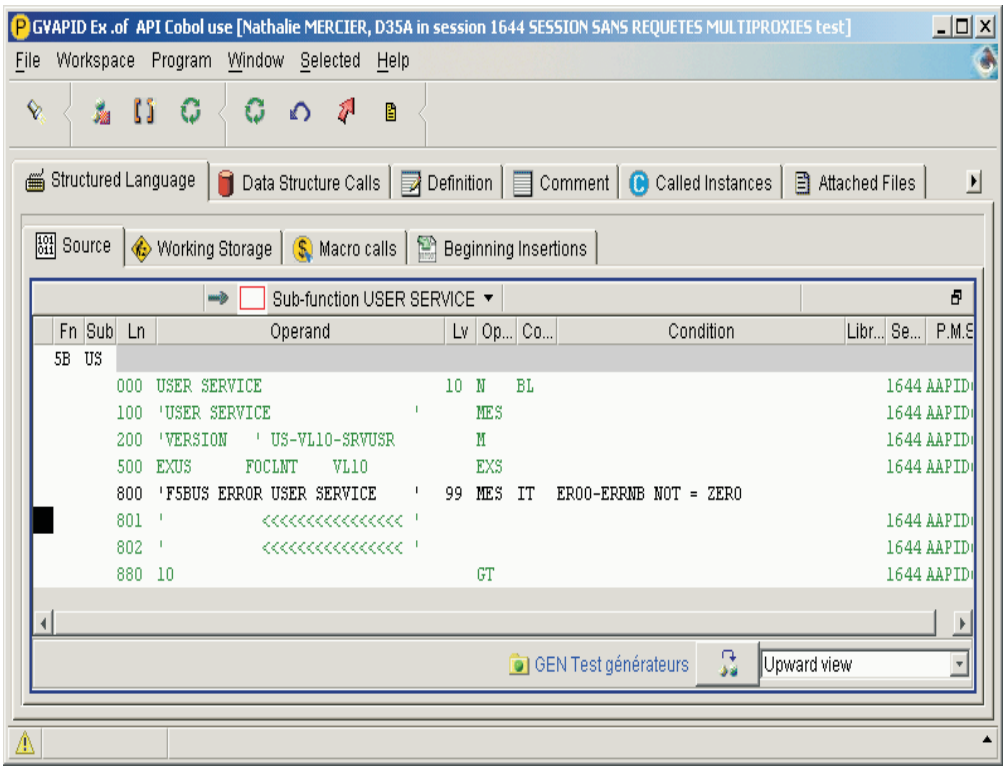
The eBusiness Application and Folder should have been defined in the work areas to be able to run correctly.

You must initialize the structures required to ensure the smooth running of the service: the presence indicators of the Logical View structure concerned, possibly the user buffer, and the US-xxxx-SRVUSR field (xxxx being the code of the node corresponding to the service).

The contents of the US-xxxx-SRVUSR field must correspond to a service available on the node concerned.

The service return initializes the presence indicators and the structure of the Logical View or the structure of error messages.

To test whether the service is running correctly, check that the number of errors (ERRNB field in the generated program) is not superior to 0 at the service return.



Initialization and termination services

Thanks to these services, you can define a consistency work unity. To do so, you must call an Initialization Server used to open the work unity, and a Termination Server used to close the unity.

To specify an Initialization Server and a Termination Server, do as follows:

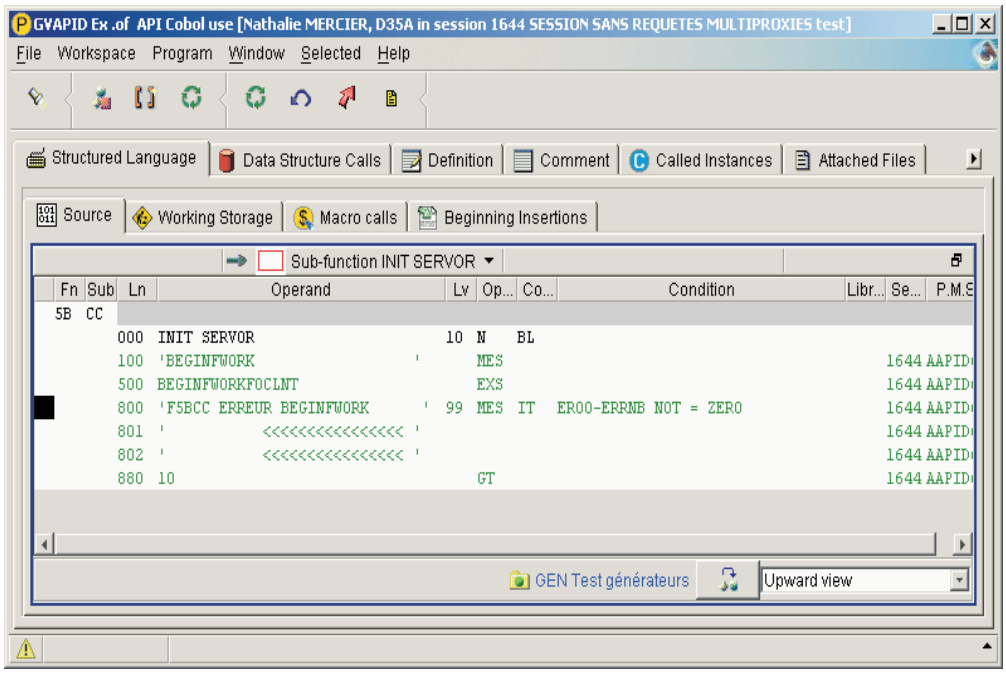
```
EXS BEGINFWORKPARAM1
EXS ENDFWORK  PARAM1
```

Où

```
   PARAM1  = XXXXXX  Program code of the Folder
```

The first service calls the Initialization Server associated with the Folder and the second one calls the Termination Server.

To test the service execution, check that the number of errors (ERRNB field in the generated program) is not superior to 0 at the service return.


GVAPID Ex .of API Cobol use [Nathalie MERCIER, D35A in session 1644 SESSION SANS REQUETES MULTIPROXIES test]
File Workspace Program Window Selected Help
Structured Language
Data Structure Calls
Definition
Comment
Called Instances
Attached Files
Source
Working Storage
Macro calls
Beginning Insertions
Sub-function INIT SERVOR
Fn Sub Ln Operand Lv Op... Co... Condition Libr... Se... P.M.S
5B CC
000 INIT SERVOR 10 N BL
100 'BEGINFWORK ' MES 1644 AAPID
500 BEGINFWORKFOCLNT EXS 1644 AAPID
800 'F5BCC ERREUR BEGINFWORK ' 99 MES IT ERROO-ERRNB NOT = ZERO 1644 AAPID
801 ' <<<<<<<<<<<<<<<< ' 1644 AAPID
802 ' <<<<<<<<<<<<<<<< ' 1644 AAPID
880 10 GT 1644 AAPID
GEN Test générateurs
Upward view

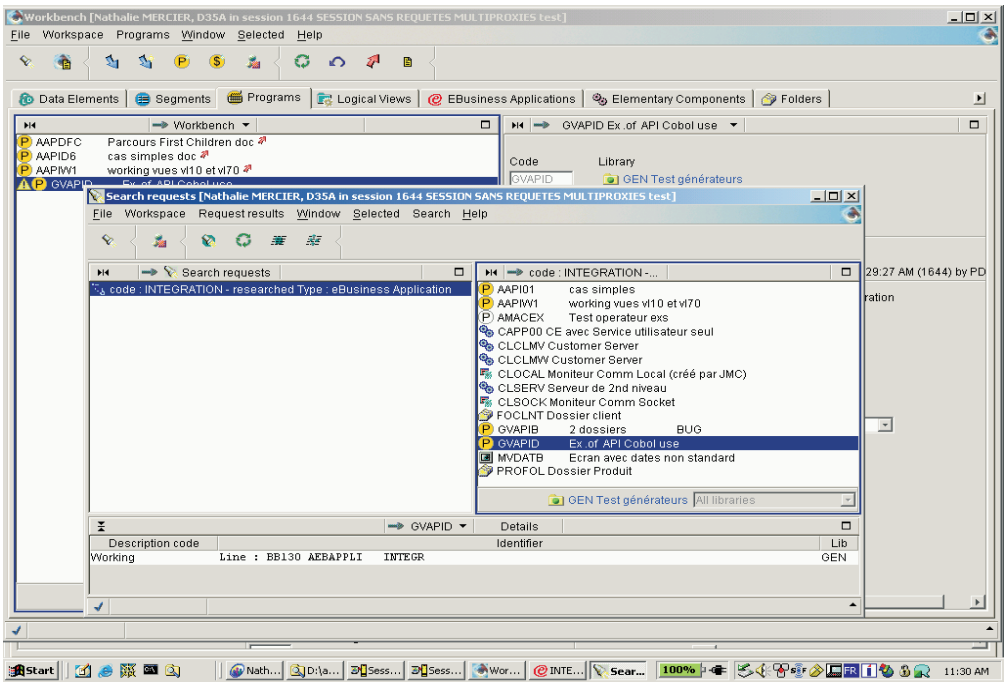
Chapter 4. Linkings

To ensure an easy use of the COBOL API, linkings are used to associate the eBusiness entities (eBusiness Application and Folder) and the application program in which they are called, on the work areas lines of 'A' type.

Therefore, linkings can be used between:

- eBusiness Application and Program, Elementary Components, Screen.
- Folder and Program, Elementary Components, Screen.

The search of linkings is possible on both the local mode and the Server mode.



Part Number: DDAPI000351A - 6556

Printed in USA