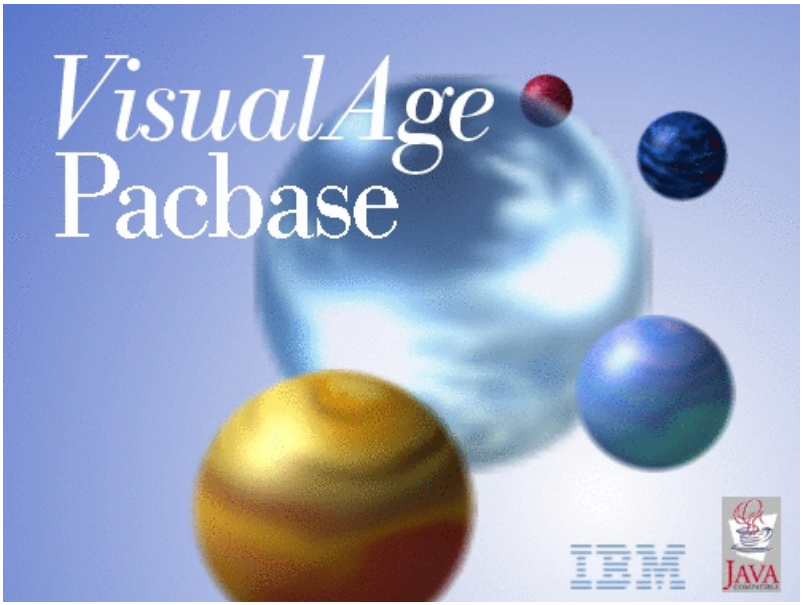
VisualAge Pacbase

The Developer's Procedures Windows 2000 or NT Server

Version 3.5



VisualAge Pacbase

The Developer's Procedures Windows 2000 or NT Server

Version 3.5

Note

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

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.

Ninth Edition (December 2010)

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. General Introduction to the Batch Procedures

Foreword

This manual documents the batch procedures that all the product users are likely to use.

These procedures first include all standard procedures dedicated to updating, generating, printing, and extracting.

They also include the procedures dedicated to the following functionalities:

- Personalized extraction and automated documentation,
- Integrity checks on Methodology occurrences (associated with the VA Pac WorkStation's Pacdesign module for SSADM and YSM),
- Pac/Impact.

Overview of the Procedures

Batch processes are grouped into procedures. The objective of the following chapters is to present each of the procedures that are likely to be used, and to specify their execution conditions.

The following elements are included for each procedure:

- a general introduction including:
  - the Execution Conditions,
  - operations to be performed in case of Abnormal Executions.
- the description of the User Input, Processes and Results obtained, possibly including use recommendations.
- the Description of Steps.

To use a procedure on a given Database, the user must have the corresponding authorization.

Each user has:

- a general level of authorizations to the batch procedures,
- a specific authorization level per Database.

User authorizations are defined in the Administration Database.

NOTE

The definition and the execution mode of a procedure are described in the Installation Guide, chapter 'Installation of Server Environment', sub-chapter 'Installation of System Environment', paragraph 'An element of the System: the procedure'.

User Identification '*' Line

Batch procedures which access the Databases require a user identification ('*'-type) line at the beginning of user input to identify the user as well as the Library and session in which he/she wishes to work.

Some information entered on this line is the same as that entered on the Sign-on screen. It is thus possible to check if the user's commands are compatible with his/her authorizations.

Before running any batch procedure, the user must make sure he/she has the adequate authorization level.

Position	Length	Value	Meaning
2	1	'*'	Line code
3	8	uuuuuuuu	User code
11	8	pppppppp	Password
19	3	bbb	Library code
22	4	nnnn	Session number (space if current session)
26	1		Session type
		' '	Current session (if session number set to space)
		'T'	Test frozen session
27	1		With the UPDT procedure in case of multiple deletion:
		'N'	Print all transactions, including generated transactions (default option)
		'O'	Print transactions entered by the user and erroneous generated transactions
		'E'	Print erroneous transactions only
			The 2 following fields must be valued for all the extraction procedures which generate update transactions which will modify a Library/session under DSMS control (you can also indicate them on the UPDT '*' line,

Position	Length	Value	Meaning
40	3		Product code (3 character-code),
43	6		Change number (6 character-code, non-significant zeros must be entered),
			These two codes will appear in the Journal after the execution of UPDT
49	1		Transfer of Entity Lock:
		blank	Replacement of the user code which locks the entity with the user code of the '*' line
		'1'	New entities created from the extracted entities are not locked after the execution of UPDT
		'2'	The user code which locks the entity is kept
50	1		Transfer of the password on the extraction procedures, on the '*' line of output transactions
		blank	The password is not transferred into the output file,
		'1'	The password is transferred, (Note : for EXTR, the '*' line is transferred into the output file only if you have entered a 'C' in Column 1)
67	1	'N'	This value is systematically set by the extractors. It indicates that the extracted transactions come from a consistent environment and that updates are always performed with a warning in case of error during a control. Keeping the data consistency triggers, among other things, the inhibition of the lowercase/uppercase conversion and the acceptance of Data Elements formats greater than 999 characters.

Access Authorizations

An '*' line with a user code and password is required by all procedures.

The Administrator manages the user access authorizations on batch procedures via the Administrator workbench.

Abnormal Endings

Abends may occur during the execution of a batch program. Input-output errors on the system or Database files cause a forced abnormal end with a return code '12', described by a message in the .Log file of the procedure.

When an abend occurs, you must find the error message. This message is displayed in the following format:

```
PROGR : pppppp  INPUT-OUTPUT ERROR : FILE ff  OP : oo
STATUS : ss
END OF RUN DUE TO PROVOKED ABEND
```

This message is displayed if you have previously set the 'BVPTrace' variable to 'YES' in Init.vbs or in the procedure's startup script.

In most cases, examining the status and type of operation enables you to find the cause of the abnormal execution.

The summary table below lists the most common values for the status and type of operation.

Code	Operation
W	WRITE
RW	REWRITE
RU	READ UPDATE
OP	OPEN
CL	CLOSE
D	DELETE
R	READ
P	START
RN	READ NEXT

Status	Message
10	End of file
21	Sequence error
22	Duplicate key
23	Record not found
24	Boundary violation
30	System error
34	Boundary violation (sequential)

Status	Message
35	File not found
46	No current record (for a READ). The error occurs when the previous operation is an abended START, which left the pointer not defined.
48	Attempt at writing on a file which is not open or on a sequential file open in I/O.
92	Logical error (For example, the opening of a file which is already open)
93	File still open in on-line mode
95	Invalid or Incomplete file

When there is no such message, and if the type of ABEND generated directly reports a problem in the product programs, contact the product support at IBM. KEEP ALL LISTINGS that may be necessary to analyze the problem.

If the error is not an input-output error on a Database file, the following message is displayed:

```
Run Time Error nnn
```

where nnn is the error number.

The Run Time Error 013 is the most frequent. It indicates that the procedure did not find an input file.

The next subchapter contains the list of the most frequent errors. Each Run Time Error is briefly described.

If the Run Time Error is not in the following list or if its associated description is not explicit enough and if the error directly involves the system programs, you must contact the Hot Line and keep all listings which might be useful in solving the problem.

List of Run-Time Errors

This list is a reminder of the most common errors and their meaning.

```
Number   Meaning

------   -------
004      Invalid file name
005      Invalid device specification
007      No more disk space
009      Directory full or does not exist
013      File not found
026      Block I-O error
```

```
027      Device not available
028      Disk space exhausted
033      Physical I-O error
105      Memory allocation error
116      Cannot allocate memory
135      File not found
150      Program abandoned on user request
157      Not enough program memory:  object file too big
         to load
170      System program not found
173      Called program file not found
188      File name too long
198      Not enough program memory: object file too large
         to load
207      Machine does not exist on the network
208      Network communication error
209      Network communication error
221 !
222 !>   Error during a SORT
223 !
```

Procedures Error Management

If an error is detected at the end of a procedure, the procedure stops with a return code other than zero. This return code can be retrieved via the "Return" variable right after the command which submits the procedure.

This prevents the execution of the next procedures if various procedures are executed in sequence.

Server startup and shut down

The Listener must be installed in Service NT mode.

Workstations and terminals can then connect to VisualAge Pacbase.

Via the 'Start [DBase_name] Database Service] shortcut located under the programs group [VisualAge Pacbase 3.5 Server] in the 'Start' menu, you can start the listener on the [DBase_name] Database.

Via the 'Stop[DBase_name] Database Service] shortcut located under the programs group [VisualAge Pacbase 3.5 Server] in the 'Start' menu, you can stop the listener on the [DBase_name] Database.

The operating parameters of the listener are defined in the "Server.wsf" procedure.

Connection of a 3270 Emulator

You can access an online server in a 'dumb terminal' mode via a 3270 emulator.

The emulator must be configured accordingly, i.e. you must indicate:

- the IP address of the machine where the on-line server is installed,
- the on-line server port number, chosen at installation time when the Database is created.

The code page of the emulator must be valorized according to the Database language code:

- code page 1147 for a French Database,

- code page 1146 for an English Database.

These code pages are automatically set in the "Server.wsf" procedure when the online server is started up.

Chapter 2. Generation and Printing

GPRT - The Generation/Print Procedure

GPRT - Introduction

The Generation and Printing procedure, GPRT, has a two-fold purpose:

- To print documentation using data contained in the Database, and
- To generate Programs, Screens, eBusiness components, Database descriptions, Data Structures and error messages.

This procedure does not affect the Database. Therefore, it may be executed while the files are open to on-line use.

However, if the generation-print requests submitted on line (+AG) are to be included, the files of the Development Database must be closed. The procedure invalidates the print requests submitted on line, therefore the file must be accessible for update.

GPRT calls only one program (BVPACB), which is used as a monitor which calls the different programs that make up the procedure.

All the programs that make up the procedure are thus considered as sub-programs of this monitor, with which they communicate via a communication area and specific return codes.

To process all the various user requests, this procedure is broken down into 'sub-chains' whose purpose is to process, in an integrated manner, the preparation of the generation-print requests for the types they manage.

Following the execution of the two general programs that are common to all chains (BVPACA10 and BVPACA20), the sub-chains are activated, if appropriate, in the following order:

- Database Blocks,
- SQL Database Blocks,
- COBOL programs,
- On-line Screens,
- Client Screens,
- Server Screens,
- eBusiness Error Messages,
- Error Messages and Dialog Windowing,

- Personalized Documentation Manager,
- Batch programs,
- Specifications Dictionary.

The files which contain the 'generated source code' (ready to be compiled or to be stored in an Assembler or Source Library) are concatenated into a single physical file that will be used in the following step.

The User Error Message file is updated using the LG-suffixed, and is retrieved into a GL-suffixed file. This file is used to update the User Error Message file. It is used in input to the EMLD or EMUP procedures. In addition, these elements are printed in the IL-suffixed file.

The installed procedure does not provide names for the two versions of this file. Therefore, the names must be specified when these messages are generated.

Volumes are standardly printed in an IN-suffixed file. The GN-suffixed file can also be used (record length = 265) with the 'ASA' skip character in the first position of each record when special print characteristics are needed.

The file containing the elements necessary for the windowing of OLSD applications is coded PAC7GT (record length is 260). Its name must be specified in the generation request.

RPPz Utilities:

With the C9 option, this procedure generates the entities which contain micropatterns.

It generates the micropatterns for the -W lines (types I, E, S, F) which do not come from Macros.

The generation is carried out without line numbers.

The right part (columns 73 to 80) is generated as a blank, except for the function tag lines where lvnn is generated (with nn = level of the function/subfunction).

Execution conditions

The files can remain open, except if the generation-print requests have been submitted on line via the ' +AG' command. In this case, the files of the Development Database must be closed.

Abnormal execution

Refer to chapter 'Overview', subchapter 'Abnormal Endings' in the Administrator's Procedures' manual.

GPRT and the SCM module

If the SCM module is available on the site, the generation may create transactions in the QJ file, an archived journal file which contains generated COBOL information such as the Pacbase-constants.

Only the entities defined in an SCM environment and generated from a production session or the current session are recognized to complete QJ.

The QJ transactions can be automatically transferred into the Development Database(s) after the generation, with options specified as parameters in the generation step. The files of the Development Database(s) can remain open.

So the generated entities defined in the SCM Environments are complemented with information related to the last processing of these entities. The status of the entities generated in the current session becomes 'production in wait'.

If errors are found, they are stored in the QJ file. They are printed in output of the ARPM procedure (transactions archiving), and the erroneous transactions are restored in the QJ file in order to be processed again.

GPRT - User Input / Results

Input

The GPRT procedure requires the following input:

- a line which identifies the user and the generation-print context,
- one line per generation or print request,
- an optional line ('+AG') which takes into account the requests already submitted on line.

Any other type of transaction is ignored.

Results

There are two types of results:

- A report which lists the requests,
- All the printings requested.

Requests are sorted by user/library and are preceded by a 'banner' (title page).

Note

This procedure does not increment the session number.

GPRT - Generation / Print Commands

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
2	1		Line code
		'Z'	Default value
3	2		Processing sequence order
			This field is used to specify the sequence in which print requests are processed and printed.
5	4		GENERATION-PRINT COMMANDS
			NOTE: Input of the entity code is required or optional depending on the command. The following indicators describe the various options:
			(A) Required entity code input (Batch column 9).
			(B) Optional entity code input. If omitted, all the occurrences of the entity type are listed in the user's hierarchical view. field.
			(C) Entity code input not allowed. All occurrences of the entity type are listed in the user's hierarchical view.
			(D) A blank line may be requested. Type an asterisk in the CONTINUATION OF REQUEST INDICATOR (C) field and press the ENTER key. The options for each command are listed below. This corresponds to batch columns 31 to 80 incl.
			NOTE: Each command may require additional information. The following list identifies these input fields by code.
			(1) SEL: _ Limit the list by keyword type: enter 'M' for explicit, 'L' for implicit, or blank for both. In batch mode, enter this value in column 30. See also SELECTION OF KEYWORD TYPE.
			(2) Same as above plus a following line on which a user may enter one or several keywords. This appears as a continuation line in on-line mode and corresponds to batch columns 31 to 80.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			(3) FORMAT: _ A format may be specified: enter 'I' for internal, 'E' for input, or 'S' for output. Enter these values in column 17 in batch mode. A blank is also valid and means that the default value is desired. See also TYPE TO SELECT.
			(4) CCF:_ CCB: The code of the control card in front of program and in back of program, respectively. Enter these codes in columns 19 to 22 in batch mode. These codes must be consistent with the codes displayed on the Dialog Definition screen.
			(5) CCF:__ CCB:__ The code of the control card in front of program and in front of map, and the code of the control card in back of program and in back of map, respectively. The user can override the default control cards. These codes should be consistent with the values on the Dialog Definition screen. In batch mode, use columns 19 to 22.
			(6) TYPE: __ The user enters the selected type which should be consistent with the corresponding field on the Definition screen of that entity type. In batch mode enter the type in columns 17 and 18.
			(7) PRINT DOCUMENT Y CHAP/SUBCHAP AND CODE:_ __ __ Specify the chapter and/or subchapter. Enter 'C' for chapter followed by the chapter code, or 'S' for subchapter followed by the chapter and subchapter codes. In batch mode use columns 23 through 27.
			(8) ENV.:__ (CCF:__ CCB:__) For those sites that are using the SCM option, the environment may be specified. In batch mode enter the environment code in column 17 and the corresponding control cards in columns 19 through 22.
			THESAURUS
		DCK	(C) A complete Description of Keywords defined in the thesaurus which lists the SYNONYM OR DEFINITION field contents associated with each keyword.
			Note: This data being specified in Inter-Library only, this command cannot be used with the U1 option. Use the C1 or I1 option which gives the same output.
		LCK	(1) (C) A listing of all keywords defined in the thesaurus, with their synonyms. It includes the number of uses of these keywords in the Database. The information is sequenced by code.
			TEXTS
		DCT	(A) Description of selected Text.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			Note: If you enter an asterisk in the ENTITY CODE field, the Descriptions of all Text occurrences are printed, sequenced by code.
		DTT	(B) (6) Descriptions of Text occurrences sequenced by type.
		L*T	List of Texts with their paragraphs titles, sequenced by code.
		LCT	(C) List of Text occurrences sequenced by code.
		LKT	(2) List of Text occurrences whose names and/or explicit Keywords contain the Keyword(s) specified.
		LNT	List of Text occurrences sequenced by name.
		LTT	(6) List of Text occurrences sequenced by type.
			DOCUMENTS (PDM)
			Note: DOCUMENT entity = VOLUME entity in the VA Pac character-mode interface.
		DCV	(B) Printing of the Description of the Document whose code is entered in the Entity field. When this code is not entered, the Descriptions of all the Documents are printed, sequenced by code.
		FLV	(C) (D) (4) Flow control for Documents.
		LCV	(C) List of Documents sequenced by code.
		LKV	(C) (2) List of Documents selected according to the keyword(s) entered on the continuation line.
		LNV	(C) (2) List of Documents sequenced by name.
		PCV	(B) (D) (7) Printing of the contents of the Document whose code is entered in the ENTITY CODE field. When this code is not entered, the contents of all the Documents are printed, sequenced by code. For local printing in RTF format, the Document must be generated with the C2 option. Selective Printing is documented in the 'Personalized Documentation Manager' manual, chapter Access Commands, subchapter 'Generation-Printing'.
			ELEMENTS AND PROPERTIES
		DCE	(B) A complete description of the defined Element(s). The information is sequenced by Element code.
			Note: to display the assigned text, use print option '2'.
		DFE	(B) A listing of the Element(s) not defined in the Specifications Dictionary, with cross-references.
		LAE	(C) List of Elements sequenced by COBOL name.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		LCE	(B) A list of defined Elements sequenced by Element code.
		LKE	(C) (2) A list of Elements and properties sequenced by keyword.
		LNE	(C) A list of Elements and properties sequenced by name.
		LXE	(C) A list of defined Elements and properties which are not used.
			DATA STRUCTURES
		DCD	(B) A complete Description of the Data Structure(s). This includes cross-references to Programs and Screens and a list of associated Reports and Segments. The information is sequenced by Data Structure code.
			Note: To get the associated text use print option '2'.
		FLD	(C) (D) (4) This command is used to specify the job card and end- of-job delimiters: flow control of Data Structures.
			Use the continuation line to define user parameters on the control cards.
		GCD	(A) Generates a COBOL description (COPY book) of the Data Structure.
			Upon generation, a Segment can contain up to 9999 Data Elements. An error message is displayed in the generation report if this number is exceeded (for more details on generation, refer to the 'Data Dictionary' manual).
			C3 : Generation of comments which will be used by VA Pac Connector (an eBusiness tool).
			C4 : All the calls to the DATA and DATASQ P.I.As. will be ignored
		LCD	(C) A list of Data Structures sequenced by code.
		LED	(A) List the error messages defined for the Data Structure and for each Segment. This list only includes messages that have already been generated.
		LKD	(C) (2) A list of the Data Structures whose names and/or explicit keywords contain the keyword(s) specified.
		LND	(C) (2) A list of the Data Structures sequenced by name.
		LOD	(C) A list of Data Structures sequenced by external name.
		LPD	(C) A list of Data Structures sequenced by Program external name.
		LTD	(C) A list of Data Structures sequenced by type.
			SEGMENTS AND LOGICAL VIEWS

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		DCS	(B) (D: with input of the entity code) (3)
			You must enter the Data Structure code in the ENTITY field, and the last 2 characters of the Segment code(s) on the continuation line (only one continuation line is possible, i.e. 25 Segments).
			A complete Description of the Segment(s). This includes cross-references to Programs and Screens for the Data Structure and to all entities for the Segment(s) and a list of associated Reports and Segments. For Segments defined as tables (Pactables function), a list of subschemas and subsystems is printed.
			Note: To get the associated text for both the Segment and the Data Structure, use print option '2'.
		LCS	(C) List of Segments sequenced by code.
		LKS	(C) (2) List of Segments whose names and/or explicit keywords contain the keyword(s) specified.
		LNS	(C) List of Segments sequenced by name.
			INPUT AIDS
		DCI	(C) A complete description of the Input Aid(s) including a list of uses of the Input Aid(s) in other entities. The information is sequenced by the PIA code.
		LCI	(C) A list of Input Aids sequenced by the PIA code.
		LKI	(C) (2) A list of the Input Aids whose names and/or explicit keywords contain the keyword(s) specified.
		LNI	(C) (2) A list of the Input Aids sequenced by name.
		LXI	(C) List of all cross-references (PIA calls) as defined on the PIA Description screen sequenced by the value of this field.
			DATABASE BLOCKS
		DTB	(B) (6) Description(s) of Database Blocks of the type specified including cross-references to other Blocks and Screens.
			Note: To get the associated text, use print option '2'
		FLB	(C) (D) (4) (8) This command is used to specify the job card and end- of-job delimiters: Flow control of the Database Block.
		FLS	(C) (D) (4) (8) Same as FLB for Relational/SQL Blocks.
			Use the continuation line to define user parameters on the control cards.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		GCB	(A) (D) (4) Generate a DDL description of the Database Block specified (including 'DB'-type Blocks for DB2).
			Use the continuation line to define the user parameters on the control cards.
			Upon generation, the Segments called in a Block can contain up to 999 Data Elements each. An error message is displayed in the generation report if this number is exceeded.
		GSQ	(A) (D) (4) Generates the SQL DDL for the Relational/SQL Database Block specified. Use the continuation line to define the user parameters on the control cards.
			Upon generation, the Segments called in a Block can contain up to 999 Data Elements each. An error message is displayed in the generation report if this number is exceeded.
		LCB	(C) List of Database Blocks sequenced by code.
		LEB	(C) List of Database Blocks sequenced by external name.
		LES	(C) List of SQL objects sequenced by external name.
		LKB	(C) (2) A list of the Database Blocks whose names and/or explicit keywords contain the keyword(s) specified.
		LNB	(C) (2) A list of Database Blocks sequenced by name.
		LTB	(6) A list of Database Blocks whose Block type have been defined with the specified value.
		LTS	(C) A list of SQL objects sequenced by code.
			* FOLDERS, FOLDER VIEWS, BUSINESS COMPONENTS, * C/S SCREENS (TUI CLIENT COMPONENTS) * SCREENS, DIALOGS.
		DCO	(A) Complete Screen Description including Dialog Complement and uses in other Screens.
		DGC	(A) Complete Description of a Pacbench C/S Screen.
		DGS	(A) Complete Description of a Pacbench C/S Business Component.
		DSO	(A) Description of the selected Screen.
		FGC	(C) (D) (4) (8) Flow control for Pacbench C/S Screens.
		FGE	(C) (D) (4) Flow control for Pacbench C/S error messages.
		FGS	(4) Flow control for Server Component.
		FLE	(C) (D) (4) Flow control for Dialog error messages.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		FLO	(C) (D) (4) (8) Flow control for Screens.
		FME	(4) Flow control for eBusiness Error messages.
		FMS	(4) Flow control for Server.
		FSO	(C) (D) (4) (8) Flow control for source Screen.
		GCO	(A) (D) (5) Generates a COBOL Description of the Screen specified.
			Upon generation, the Segments called in a Screen can contain up to 999 Data Elements each. An error message is displayed in the generation report if this number is exceeded.
			Note: If you have specified the 'COBOL formatting option' on the Library, up to 150 lines are possible in a COBOL paragraph (between two periods).
		GGC	(A) (D) (5) Generates a C/S Screen (TUI Client Component).
			Upon generation, the Segments called in a C/S Screen can contain up to 999 Data Elements each. An error message is displayed in the generation report if this number is exceeded.
		GGS	(A) (D) (5) Generation applicable to Business Component, Communication Monitor, Error Server, Folder.
			Upon generation, the called Segments can contain up to 999 Data Elements each. An error message is displayed in the generation report if this number is exceeded.
		GEC	(A) (D) Pacbench C/S:
			C1 : Error messages defined for the Client or Server Dialog and for each component.
			Note: In the 'LANG' field which is displayed after a transmit, you can enter the generation language (EN or FR) of the error messages. If you do not enter any language code, the messages will be generated in your assigned language. If you enter a code other than EN or FR, the messages will be generated in English.
			C2 : Error messages generated through option 1 plus documentary help messages.
			C3 : Error messages for the Dialog only.
		GED	(A) (D)
			C1 : Error messages generated for a Data Structure and for each Segment.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			Note: In the 'LANG' field which is displayed after a transmit, you can enter the generation language (EN or FR) of the error messages. If you do not enter any language code, the messages will be generated in your assigned language. If you enter a code other than EN or FR, the messages will be generated in English.
			C2 : Error messages generated through option 1 plus documentary help messages.
		GEO	(A) (D) OLSD Function:
			C1 : Error messages defined for the Dialog and for each Dialog Screen.
			Note: In the 'LANG' field which is displayed after a transmit, you can enter the generation language (EN or FR) of the error messages. If you do not enter any language code, the messages will be generated in your assigned language. If you enter a code other than EN or FR, the messages will be generated in English.
			Note: The header of the generation report displays a summary report of the errors detected during generation and the list of the Screens which have not been generated.
			C2 : Error messages generated through option 1 plus documentary help messages.
			C3 : Error messages for the Dialog only.
			C4 : Generation of a file which contains the data required for the Screen revamping with the Dialog Web Revamping module. If the Screen code includes special characters, an error is generated.
			Note: If a Segment/Screen suffix is entered on the continuation line of one of the preceding commands, error messages are generated/printed only for the selected Segment/Screen.
		GEF	(A) Generation of error messages for a C/S Folder.
			Note: In the 'LANG' field which is displayed after a transmit, you can enter the generation language (EN or FR) of the error messages. If you do not enter any language code, the messages will be generated in your assigned language. If you enter a code other than EN or FR, the messages will be generated in English.
		GEI	(A) Generation of error messages for INIT/TERM component.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			Note: In the 'LANG' field which is displayed after a transmit, you can enter the generation language (EN or FR) of the error messages. If you do not enter any language code, the messages will be generated in your assigned language. If you enter a code other than EN or FR, the messages will be generated in English.
		GES	(A) Generation of error messages for a C/S Component.
			Note: In the 'LANG' field which is displayed after a transmit, you can enter the generation language (EN or FR) of the error messages. If you do not enter any language code, the messages will be generated in your assigned language. If you enter a code other than EN or FR, the messages will be generated in English.
		GSO	(A) Generates source code for the selected Screen.
		GVC	(A) (D) (5) Extract a Proxy object. Applicable to Folder View, Folder and Business Component.
		GMF	(A) Generates a Folder.
		GMI	(A) Generates an INIT/TERM Server.
		GMM	(A) Generates a Communication Monitor.
		GMS	(A) Generates a Server.
			Upon generation, the Segments called in the Server can contain up to 999 Data Elements each. An error message is displayed in the generation report if this number is exceeded.
		GME	(A) Generates an Error Server.
		LCO	(C)
			List of Screens sequenced by code.
		LEC	(A) List the error messages defined for the Client Component and for each Client Screen. This list only includes messages that have already been generated.
		LEO	(A) List the error messages defined for the Dialog and for each Screen. This list only includes messages that have already been generated.
		LKO	(C) (2) List of Screens whose names and/or explicit keywords contain the keyword(s) specified.
		LNO	(C) List of Screens sequenced by name.
		LOT	(C) List of Screens sequenced by the entered Transaction code.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		LPO	(C) List of C/S Screens sequenced by external program name.
		LSO	(C) List of C/S Screens sequenced by external map name.
		LTO	(C) List of Screens sequenced by type.
			REPORTS
		DCR	(B) (D: when the entity code has been entered)
			Note: In the ENTITY field, you must enter a Report code (on 3 characters) to print the Description of one Report, or blanks to print the descriptions of all the Reports. (The commands created with versions earlier than the 3.5 V05 must be manually modified before submitting the print request).
		LCR	(C) List of Reports sequenced by code.
		LTR	(C) List of Reports sequenced by type.
		LKR	(2) A list of the Reports whose names and/or explicit keywords contain the keyword(s) specified.
		LNR	(C) List of Reports sequenced by name.
			PROGRAMS
		DCP	(B) A complete description of Program(s). The information is sequenced by the Program code.
			Note: To get the associated text, use print option '2'.
		DSP	(A) Description of the selected Program produced by Reverse Engineering.
		FLP	(C) (D) (4) (8) This command is used to specify the job card and end-of-job delimiters: Flow control for Programs.
			Use the continuation line to define user parameters on the control cards.
		FSP	(C) (D) (4) (8) This command is used to specify the job card and end-of-job delimiters: Flow control for 'reverse engineered' programs. Use the continuation line to define user parameters on the control cards.
		GCP	(A) (D) (4) Generates a COBOL description of the Program specified Use the continuation line to define user parameters o the control cards. Upon generation, the Segments of the Data Structures called in the Program can contain up to 9999 Data Elements. An error message is displayed in the generation report if this number is exceeded.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			Note: If you have specified the 'COBOL formatting option' on the Library, up to 150 lines are possible in a COBOL paragraph (between two periods).
		GSP	(A) (D) (4) Generate a COBOL description of the 'reverse engineered' Program specified. Use the continuation line to define user parameters on the control cards.
		LCP	(C) List of Programs sequenced by program code. Note: To get keywords, use print option '2'.
		LEP	(C) List of Programs sequenced by external name.
		LKP	(2) A list of the Programs whose names and/or explicit keywords contain the keyword(s) specified.
		LNP	(2) List of Programs sequenced by name.
		LTP	(C) List of Programs sequenced by type.
			METHOD ENTITIES
		DCM	(A) A complete Description of the Method entity as specified.
		DCMC	(C) A complete Description of Method Functional Integrity Constraint(s).
		DCMO	(C) A complete Description of Method Object(s).
		DCMR	(C) A complete Description of Method Relation(s).
		LCMC	(C) List of Method Functional Integrity Constraints sequenced by F.I.C. code.
		LCMO	(C) List of Method Objects sequenced by Object code.
		LCMP	(C) List of properties sequenced by Property code.
		LCMR	(C) List of Method Relations with their Functional Integrity Constraints, sequenced by Relation code.
		LKM	(C) (2) A list of the Method entities whose names and/or explicit keywords contain the keyword(s) specified.
			META-ENTITIES
		DCF	(B) A complete Definition and Description of the Meta-Entity entered in the ENTITY field. If no code is specified, all Meta-Entities are listed. The information is sequenced by code.
		DCQ	(B) A complete Definition and Description of the User Relations entered in the ENTITY field. If no code is specified, all User Relations are listed. The information is sequenced by code.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		DCY	(B) A complete Definition and Description of the Extended User Entity entered in the ENTITY field. If no code is specified, all Extended User Entities are listed. The information is sequenced by code.
		DC$	(B) A complete Definition and Description of the User Entity entered in the ENTITY field (the following form is required: DC$xx, where xx corresponds to the type of entity call).
		LCF	(C) List of Meta-Entities sequenced by code.
		LCQ	(C) List of User Relations sequenced by code.
		LCY	(A) List of Extended User Entities sequenced by code.
		LC$	(A) List of User Entities sequenced by type and code (LC$xx, xx being the type of entity call).
		LKF	(2) (C) A list of the Meta-Entities whose names and/or explicit keywords contain the keyword(s) specified.
		LKQ	(2) (C) A list of the User Entities Relations whose names and/or explicit keywords contain the keyword(s) specified.
			Note: For all printing by keyword, you can specify the TYPE OF SELECTION (BLANK, L or M) on the print line. Keywords are indicated on the continuation line sent back.
		LKY	(2) (A) A list of the Extended User-Entities whose names and/ or explicit keywords contain the keyword(s) specified.
		LK$	(2) (A) A list of the User Entities whose names and/ or explicit keywords contain the keyword(s) specified.
		LNF	(C) A list of the Meta-Entities sequenced by name.
		LNQ	(C) A list of the User Relations sequenced by name.
		LNY	(A) A list of Extended User-Entities sequenced by name.
		LN$	(A) A list of the User Entities sequenced by name.
			SHIFT TO UPPER-CASE
		UPC	This command allows for the automatic transformation of lower-case letters into upper-case letters in the printed output of the GPRT procedure.
			When the UPC command is entered, the following line is displayed:
			SHIFT TO UPPERCASE MANUAL:_ DOC:_ ERROR MESS:

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			The VA Pac user must specify to which type of GPRT output the UPC command will apply (even when only one GPRT command is validated).
			In order to do this, the value '1' must be entered in one of the three fields displayed above: in the MANUAL field for Volumes (V); in the DOC field for entity related commands; in the ERROR MESS field for the generation of error messages.
			Note: This also allows the selective implementation of the UPC command when the execution of several GPRT jobs is requested and the SHIFT TO UPPER-CASE must not apply to all of them, in which case the corresponding field(s) must be left blank.
		GUT	(A) Generate User Command.
			A GUT command is constituted of a first comment line and of one to five continuation lines which contain the data to be sent to the generation flow. There is no control on this command code.
			First line: The entity code is optional. The label is made up of two parts which can be modified. The first part (on 25 characters) is initialized to 'USER GENERATION' and the second part (on 11 characters) must be completed by the user. Only this second part is sent to the generation procedure and is displayed in the PAC7ID report.
			Continuation lines: five continuation lines are authorized. They are simply sent to the generation flow. They are found in the PAC7ID report and in the PAC7JC intermediate file. To process this type of request, you must then adapt your generation flow.
			Note: a GUT command without any continuation line nevertheless generates a line in the generation flow but with a space in the command.
			PAF TABLES FOR METHODOLOGY ENTITIES
		PCM	Description of PAF Tables for entities specific to a methodology. This command is necessarily followed by a Methodology code.
9	6		ENTITY CODE
			This field is displayed with the label 'ENTITY' on screen format options '1' and '2' of the GP screen.
			When required, the user enters the entity code which corresponds to the COMMAND FOR PRINT REQUEST.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			'PCM' COMMAND: In this field, you enter the code of the selected Methodology:
		M	Merise
		D	YSM
		A	SSADM
		O	OMT
		F	IFW
15	1		Library selection indicator
			Used to select the libraries from which the entities are to be generated and/or printed.
		C	Selected library and higher level libraries. In case of duplicates, the lines from the lower level library are taken into account.
16	1		PRINT OPTION
			In this field, you specify print options.
			There are 5 options numbered from 1 to 4, and 9. The default option is 1.
			Each option corresponds to presentation variants of lines to be printed, e.g. printing of additional information (with or without keywords, programs with or without associated texts, ...).
			The detail of each print option is given for each entity in the corresponding reference Manuals.
			Value 9 is reserved for RPPz.
17	2		Generation criteria
			Used to enter the language code for the GEx generation-print commands.
19	1		Control cards in front of programs
			Enter the one-character code that identifies the job card to be inserted before the generated program.
			Default: Code entered on the Library Definition Screen
20	1		CONTROL CARDS BEFORE MAP
			Screen and C/S Screen entities
			Option code that identifies the job card to be inserted before each generated Screen or C/S Screen map.
		$	No generation of map.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			NOTE: This field is not used in a Pacbench C/S development with specification of Folder.
			Business Component / single-view (with no specification of a Folder).
			Option code which selects the JCL lines to be inserted before the generated Services Manager. The value '$' is used to disable the generation of the Services Manager and to enable the Business Component to be generated.
21	1		CONTROL CARDS IN BACK OF PROGRAMS
			Enter the one-character code that identifies the job card to be inserted after the generated program.
			Default: Code entered on the Library Definition Screen
22	1		CONTROL CARDS AFTER MAP
			Screen and C/S Screen entities:
			Option code that identifies the job card to be inserted after each generated Screen or Screen c/s map.
		$	No generation of map.
			NOTE: This field is not used in a Pacbench C/S development with the specification of Folder.
			Business Component / single-view (with no specification of Folder):
			Option code which selects the JCL lines to be inserted after the Services Manager generated.
23	1		DOCUMENT SELECTIVE PRINT REQUEST
			Field displayed with PCV command only.
		blank	Print the whole Document (default value)
		C or 1	Print the selected chapter or level-1 section, respectively. Field used jointly with next field.
		S or 2	Print the selected subchapter or level-2 section (included in the level-1 section indicated in the following field), respectively. Field used jointly with next two fields.
24	2		Level-1 Section # / Chapter Code
		C	The value 'ZZ' is not authorized. CH/
26	2		Level-2 Section # / Subchapter Code
		C	SC/
30	1		SELECTION OF KEYWORD TYPE

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		blank	Selection on both implicit and explicit keywords.
		L	Selection on implicit keywords only.
		M	Selection on explicit keywords only.
31	50		Label continuation

GPRT - Procedure Startup

GPRT - Processing of Job Streams

If sources to be compiled are generated and if the return code of the previous step is lower than 8, the generated stream must be processed in order to compile and link edit the output sources.

Whichever the chosen solution, the generated sources must be preceded and/or followed by optional control lines.

The In Front/Back command lines are entered by the VA Pac Administrator.

These instructions are used to insert lines before or/and after the generated COBOL source (for example to separate generated programs).

You must specify the code of each line type (In Front/Back) on the Program or Library definition, or include it in the Generation request.

GPRT - Example of Generation Script (C4)

These instructions are used to execute the generation and print commands requested via the Generation Manager in the Developer workbench or via the GP screen.

The Administrator initializes, for each user, the generation scripts adapted to a VisualAge Pacbase Database.

Each user can modify ITS own Generation Script either in the Generation manager of Developer workbench (Command Lines tab) or via the GP screen in C4 view.

Generation Script (C4) : In Front lines

```
<job id=BVPGPRT>
<script language="VBScript">
Dim WshSh
Set WshSh = WScript.CreateObject("WScript.Shell")
```

```
Set WshEnv = WshSh.Environment("PROCESS")
Set Args = Wscript.Arguments
</script>
<resource id="GPRT"><![CDATA[
```

Generation Script (C4) : In Back lines

```
]]></resource>
<script language="VBScript">
Rep ="HKLM\SOFTWARE\IBM\BVP VisualAge Pacbase 3.5"
Rep = Rep & "\server"
Rep_PROC = WshSh.RegRead (Rep & "\BVP_SYS\PROC\")
USER=Args(0) 'user code
BASE=Args(1) 'external database code
WshEnv("BVP_Updtpm") = "NO"
WshEnv("BVP_Merge") = "NO"
WshEnv("BVP_Gpmon") = "YES"
WshEnv("BVP_Resource") = getresource("GPRT")
PROC=Rep_PROC & "\gprt.wsf"
CMD=chr(34) &PROC & chr(34)&" " &BASE & " " & USER
RetGprt = WshSh.Run(CMD , 1 , TRUE )
</script>
</job>
```

In these IN BACK lines, the following environment variables must be set to execute PACAGP:

BVP_Merge is used to merge into GPRTOM the files generated by GPRT.

BVPACAGP is used to execute the PACAGP specific processing.

To use SCM, the BVP_Updtpm environment variable, which starts the UPPM (update procedure) execution, must be set in the In Back lines.

To visualize the generated GPRT-type files in the proper format, in the Workstation (GPMON module), the following parameter must be set in the In Back lines:

WshEnv("BVP_Gpmon") = "YES"

GPRT - Specific Processes

Interface with Workbench

The purpose of this interface is to split into distinct files the sources of the programs, screens or 'COPY' clauses generated, then to write these files in a directory specified by the user.

The 'bvpsplit' program performs this processing.

The implementation of this option can only be done by activating a command file (example BVPACAGP) in the GPRT procedure.

This option also requires the definition of BEFORE lines for the VisualAge Pacbase entities to be processed, in the Generation manager.

Definition of command lines - BEFORE (Manager)

In order to allow the 'bvpsplit' program to split the source files produced by the generation, you must insert BEFORE lines which contain the following elements:

- Character strings specific to these lines
- Name of file to produce
- File extension
- Directory where the file will be copied

The first BEFORE Command line must contain :

```
*******  Delimiter for bvpsplit, between column 1
         and 7 only
filename Name of the file to be produced
ext      Extension, on 3 characters max.
```

This information must be separated by a blank. For example:

```
******* MYPROG CBL
```

The second BEFORE Command line must contain:

```
*&&&&&*  Delimiter for bvpsplit, between column 1
         and 7 only
path     File directory.
         This directory must exist and must be
         accessible via the GPRT procedure.
```

The information is separated by a blank, for example :

```
*&&&&&* S:\COBOL\2592T\USERC01
```

That is, for the BEFORE lines of a program, for example :

The user generates from frozen sessions and wishes to recover his generated programs under the form 'external_name'.CBL, in the (network) directory

S:\COBOL\'session'\'user_code'.

The control lines are defined via the Administration Workbench ('D' defines the CARD BEFORE and 'W' the card code):

```
*+++++++ <External name of generated program or block> cbl
*&&&&&* S:\cobol\<Session number>\<User code>
```

These BEFORE lines must then be called ('W' code in the example) in the entities to be generated.

Implementation in the GPRT procedure

In the GPRT start-up file, the BVPACAGP environment variable must contains the path of the commands file executed after the generations/prints (GPRT procedure).

The 'bvpsplit' program must then be called in the BVPACAGP command file, by indicating the number of the job and the generation directory of the user.

BVPACAGP must therefore contain the BVPSPLIT execution with the generated sources directory as the argument.

Processing and error messages

The 'bvpsplit' program processes all the GPRTO* files in output of the GPRT procedure, in the specified directory.

An execution report is output in the user directory (Rep_USR by default).

Example of BVPACAGP script : .vbs

```
' ====================================
' VisualAge Pacbase : BVPACAGP.vbs
' ====================================
' Args(0) = User code
' Args(1) = Job number
' Args(2) = User directory

Set WshShell = Wscript.CreateObject("Wscript.Shell")
Set Args = Wscript.Arguments
Set WshEnv = WshShell.Environment("PROCESS")
Set WshVolEnv = WshShell.Environment("VOLATILE")
MyProc =  Wscript.ScriptName
UserDir = Ucase(Args(2))

' starting BVPSPLIT
EXESPLIT =  chr(34) & "BVPSPLIT" & chr(34) & " " _
              &  chr(34) & UserDir & chr(34)
Return = WshShell.Run(EXESPLIT , 1 , TRUE )
Set Args = Wscript.Arguments
Wscript.Quit (Return)
```

Example of BVPACAGP : .cmd

```
REM ====================================
REM VisualAge Pacbase : BVPACAGP.cmd
REM ====================================
REM %1 = User code
REM %2 = Job number
REM %3 = User directory

echo %0 command procedure > %3/BVPACAGP.LOG
echo ---------------------------------- >> %3/BVPACAGP.LOG
echo
echo User code = %1 >> %3/BVPACAGP.LOG
echo Job number = %2 >> %3/BVPACAGP.LOG
echo user directory = %3 >> %3/BVPACAGP.LOG
echo

echo execution : BVPSPLIT %3 >> %3/BVPACAGP.LOG
BVPSPLIT " %3 "
if ERRORLEVEL 1 GOTO ERR
GOTO END
:ERR
echo error in executing BVPSPLIT >> %3/BVPACAGP.LOG
EXIT 1
:END
EXIT 0
```

GPRT - Description of Steps

Generation and printing: PACB

The generated documentation depends on the generation-print requests taken into account. Therefore, the volume of the generated documentation and of the temporary files is extremely variable.

Banners at the beginning and at the end of user documentation, which display the user code, facilitate the identification of their authors.

All programs, screens, Database Blocks, etc., which might be generated, are retrieved on GPRTOx files.

Some programs called by the Monitor can send specific return codes:

- BVPACA10 (Retrieval of Transactions) :

  0 : OK

  2 : OK with presence of the '+AG' command

  8 : No request.

  In this case, the procedure stops running.
- BVPACB31 (SQL generation):

  8 : Error detected during generation.
- Extractors or generators (30 or 40):

  0 : OK - No generation

  4 : OK - Generation

Other : Errors

- BVPACW10 (configuration management support)

  0 : OK

  2 : No processing

  4 : at least one parameterizing error detected.

  8 : at least one context error detected.

  This step sends a general return code.

Code	Label
4	OK with generation of source code
6	OK with generation of source code and Personalized Documentation or error messages
8	OK with generation of Personalized Documentation or of error messages
10	OK without generation
12	Input-Output error
16	Sort error

GPRT - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'    - GENERATION (IN INTERNAL READER) AND PRINTING -
'
' ------------------------------------------------------
'
' IN ADDITION TO THE GENERATED ENTITIES, THE FILE MUST
' CONTAIN THE JCL REQUIRED TO COMPILE THEM,
' USING THE BEGINNING/END OF JCL JOB STREAM OPTIONS AND
' THE BEFORE/AFTER PROGRAM OPTIONS.
'
' THE GENERATION AND PRINTING PROCEDURE, GPRT, HAS A
' TWO-FOLD PURPOSE:
'   . TO PRINT DOCUMENTATION USING  DATA CONTAINED IN THE
'       DATABASE, AND
'   . TO GENERATE PROGRAMS, SCREENS, DATABASE
'      DESCRIPTIONS DATA STRUCTURES,  AND ERROR MESSAGES.
'
' ------------------------------------------------------
'
<job id=GPRT>

<script language="VBScript">
Dim MyProc
MyProc = "GPRT"
</script>
<script language="VBScript" src="INIT.vbs"/>
```

```
<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If


Dim CodLang
If base = "ADMIN"  Then
Call Msg_Log (Array("1028",base))
Wscript.Quit (0)
Else
 CodLang = WshShell.RegRead (Rep_SYS & "\GENLANG")
End If


 If Not FSO.FileExists( Rep_AJOURNAL & "\QJ") Then
 Call Msg_Log (Array("1022" , "PCMINI"))
'-----------------------------------
 WshEnv("PAC7QJ") =  Rep_AJOURNAL & "\QJ"
 Call RunCmdLog ("BVPCMINI")

 Call Err_Cod(Return , 0 , "PCMINI")
 End if

 If Not FSO.FileExists( Rep_ABASE & "\GK") Then
 Call Msg_Log (Array("1022" , "PTUIGK"))
'-----------------------------------
 WshEnv("PACGGK") =  Rep_ABASE & "\GK"
 Call RunCmdLog ("BVPTUIGK")

 Call Err_Cod(Return , 0 , "PTUIGK")
 End if



Call Msg_Log (Array("1022" , "PACB"))
'-----------------------------------
WshEnv("PAC7QJ") =  Rep_AJOURNAL & "\QJ"
WshEnv("PAC7AE") =  Rep_SKEL & "\AE"
WshEnv("PAC7AJ") = Rep_JOURNAL & "\AJ"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PAC7AY") = Rep_BASE & "\AY"
WshEnv("PAC7LB") = Rep_BASE & "\LB"
WshEnv("PACGGK") = Rep_ABASE & "\GK"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7GS") = Rep_BASE & "\GS"
WshEnv("PAC7SC") = Rep_SKEL & "\SC" & CodLang
WshEnv("PAC7SF") = Rep_SKEL & "\SF" & CodLang
WshEnv("PAC7SG") = Rep_SKEL & "\SG" & CodLang
WshEnv("PAC7SN") = Rep_SKEL & "\SN" & CodLang
WshEnv("PAC7SR") = Rep_SKEL & "\SR" & CodLang
WshEnv("PAC7SS") = Rep_SKEL & "\SS" & CodLang
```

```
WshEnv("PAC7SL") = Rep_SKEL & "\SL" & CodLang
WshEnv("PAC7ME") = Fic_Input
Call BvpEnv("PACB","PAC7BM",Rep_TMP & "\WBM.tmp")
Call BvpEnv("PACB","PAC7DG",Rep_TMP & "\WDG.tmp")
Call BvpEnv("PACB","PAC7EB",Rep_TMP & "\WEB.tmp")
Call BvpEnv("PACB","PAC7EE",Rep_TMP & "\WEE.tmp")
Call BvpEnv("PACB","PAC7EG",Rep_TMP & "\WEG.tmp")
Call BvpEnv("PACB","PAC7EI",Rep_TMP & "\WEI.tmp")
Call BvpEnv("PACB","PAC7EN",Rep_TMP & "\WEN.tmp")
Call BvpEnv("PACB","PAC7EP",Rep_TMP & "\WEP.tmp")
Call BvpEnv("PACB","PAC7EQ",Rep_TMP & "\WEQ.tmp")
Call BvpEnv("PACB","PAC7ER",Rep_TMP & "\WER.tmp")
Call BvpEnv("PACB","PAC7EV",Rep_TMP & "\WEV.tmp")
Call BvpEnv("PACB","PAC7EW",Rep_TMP & "\WEW.tmp")
Call BvpEnv("PACB","PAC7OB",Rep_USR & "\GPRTOB.txt")
Call BvpEnv("PACB","PAC7GB",Rep_TMP & "\GPRTGB.tmp")
Call BvpEnv("PACB","PAC7OD",Rep_USR & "\GPRTOD.txt")
Call BvpEnv("PACB","PAC7GD",Rep_TMP & "\GPRTGD.tmp")
Call BvpEnv("PACB","PAC7OE",Rep_USR & "\GPRTOE.txt")
Call BvpEnv("PACB","PAC7GE",Rep_TMP & "\GPRTGE.tmp")
Call BvpEnv("PACB","PAC7OF",Rep_USR & "\GPRTOF.txt")
Call BvpEnv("PACB","PAC7GF",Rep_TMP & "\GPRTGF.tmp")
Call BvpEnv("PACB","PAC7OG",Rep_USR & "\GPRTOG.txt")
Call BvpEnv("PACB","PAC7GG",Rep_TMP & "\GPRTGG.tmp")
Call BvpEnv("PACB","PAC7GI",Rep_USR & "\GPRTGI.txt")

Call BvpEnv("PACB","PAC7GK",RepT_USR & "\ERRGK.txt")
Call BvpEnv("PACB","PAC7GL",RepT_USR & "\ERRGL.txt")
Call BvpEnv("PACB","PAC7GM",RepT_USR & "\ERRGM.txt")

Call BvpEnv("PACB","PAC7GN",Rep_TMP & "\WXGN.tmp")
Call BvpEnv("PACB","PAC7GO",Rep_TMP & "\WGO.tmp")
Call BvpEnv("PACB","PAC7OP",Rep_USR & "\GPRTOP.txt")
Call BvpEnv("PACB","PAC7GP",Rep_TMP & "\GPRTGP.tmp")
Call BvpEnv("PACB","PAC7OQ",Rep_USR & "\GPRTOQ.txt")
Call BvpEnv("PACB","PAC7GQ",Rep_TMP & "\GPRTGQ.tmp")
Call BvpEnv("PACB","PAC7OR",Rep_USR & "\GPRTOR.txt")
Call BvpEnv("PACB","PAC7GR",Rep_TMP & "\GPRTGR.tmp")
Call BvpEnv("PACB","PAC7GT",Rep_USR & "\PAWGT.txt")
Call BvpEnv("PACB","PAC7OV",Rep_USR & "\GPRTOV.txt")
Call BvpEnv("PACB","PAC7GV",Rep_TMP & "\GPRTGV.tmp")
Call BvpEnv("PACB","PAC7G6",Rep_USR & "\GPRTG6.txt")
Call BvpEnv("PACB","PAC7DB",Rep_USR & "\GPRTDB.txt")
Call BvpEnv("PACB","PAC7IA",Rep_USR & "\GPRTIA.txt")
Call BvpEnv("PACB","PAC7ID",Rep_USR & "\GPRTID.txt")
Call BvpEnv("PACB","PAC7IK",Rep_USR & "\GPRTIK.txt")
Call BvpEnv("PACB","PAC7IL",Rep_USR & "\GPRTIL.txt")
Call BvpEnv("PACB","PAC7IM",Rep_USR & "\GPRTIM.txt")
Call BvpEnv("PACB","PAC7IN",Rep_USR & "\GPRTIN.txt")
Call BvpEnv("PACB","PAC7IO",Rep_USR & "\GPRTIO.txt")
Call BvpEnv("PACB","PAC7IW",Rep_USR & "\GPRTIW.txt")
Call BvpEnv("PACB","PAC7JG",Rep_TMP & "\WJG.tmp")
Call BvpEnv("PACB","PAC7KB",Rep_TMP & "\WKB.tmp")
Call BvpEnv("PACB","PAC7KD",Rep_TMP & "\WKD.tmp")
Call BvpEnv("PACB","PAC7KE",Rep_TMP & "\WKE.tmp")
```

```
Call BvpEnv("PACB","PAC7KF",Rep_TMP & "\WKF.tmp")
Call BvpEnv("PACB","PAC7KG",Rep_TMP & "\WKG.tmp")
Call BvpEnv("PACB","PAC7KM",Rep_TMP & "\WKM.tmp")
Call BvpEnv("PACB","PAC7KN",Rep_TMP & "\WKN.tmp")
Call BvpEnv("PACB","PAC7KP",Rep_TMP & "\WKP.tmp")
Call BvpEnv("PACB","PAC7KQ",Rep_TMP & "\WKQ.tmp")
Call BvpEnv("PACB","PAC7KR",Rep_TMP & "\WKR.tmp")
Call BvpEnv("PACB","PAC7KS",Rep_TMP & "\WKS.tmp")
Call BvpEnv("PACB","PAC7KU",Rep_TMP & "\WKU.tmp")
Call BvpEnv("PACB","PAC7KV",Rep_TMP & "\WKV.tmp")
Call BvpEnv("PACB","PAC7LG",Rep_TMP & "\NUL.tmp")
'PAC7LG not used, on default
Call BvpEnv("PACB","PAC7LI",Rep_TMP & "\WLI.tmp")
Call BvpEnv("PACB","PAC7LK",Rep_TMP & "\NUL.tmp")
'PAC7LK not used, on default
Call BvpEnv("PACB","PAC7LM",Rep_TMP & "\NUL.tmp")
'PAC7LM not used, on default
Call BvpEnv("PACB","PAC7MG",Rep_TMP & "\WMG.tmp")
Call BvpEnv("PACB","PAC7MV",Rep_TMP & "\WMV.tmp")
Call BvpEnv("PACB","PAC7SO",Rep_TMP & "\WSO.tmp")
Call BvpEnv("PACB","PAC7WA",Rep_TMP & "\WWA.tmp")
Call BvpEnv("PACB","PAC7W1",Rep_TMP & "\WW1.tmp")
Call BvpEnv("PACB","PAC7W2",Rep_TMP & "\WW2.tmp")
Call BvpEnv("PACB","PAC7W3",Rep_TMP & "\WW3.tmp")
Call BvpEnv("PACB","PAC7W4",Rep_TMP & "\WW4.tmp")
Call BvpEnv("PACB","PAC7W6",Rep_TMP & "\WW6.tmp")
Call BvpEnv("PACB","PAC7W7",Rep_TMP & "\WW7.tmp")
Call BvpEnv("PACB","PAC7W8",Rep_TMP & "\WW8.tmp")
Call BvpEnv("PACB","PAC7W9",Rep_TMP & "\WW9.tmp")
Call BvpEnv("PACB","SYSPAF",Rep_TMP & "\WPAF.tmp")
Call RunCmdLog ("BVPACB")

If Return < 10  then
Call Msg_Log (Array("1062"))
Return = 0
End if
If Return = 10  then
Call Msg_Log (Array("1063"))
Return = 0
End if

If Return > 10  then
Call Msg_Log (Array("1064"))
End if
Call Err_Cod(Return , 10 , "PACB")

If BVP_Merge = "YES" then
 Call Msg_Log (Array("1022" , "COPY in OM"))
'-----------------------------------
 If Not FSO.FileExists(Rep_USR & "\GPRTOM.txt") Then
  Set LogGen = FSO.CreateTextFile(Rep_USR & "\GPRTOM.txt", TRUE)
  LogGen.Close
 End if
 OM = FSO.GetFile(Rep_USR & "\GPRTOM.txt")
```

```
 Call CopMFil(OM , WshEnv("PAC7OP") ,OM )
 Call DelFile (WshEnv("PAC7OP"))
 Call CopMFil(OM , WshEnv("PAC7OQ") ,OM )
 Call DelFile (WshEnv("PAC7OQ"))
 Call CopMFil(OM , WshEnv("PAC7OE") ,OM )
 Call DelFile (WshEnv("PAC7OE"))
 Call CopMFil(OM , WshEnv("PAC7OR") ,OM )
 Call DelFile (WshEnv("PAC7OR"))
 Call CopMFil(OM , WshEnv("PAC7OG") ,OM )
 Call DelFile (WshEnv("PAC7OG"))
 Call CopMFil(OM , WshEnv("PAC7OV") ,OM )
 Call DelFile (WshEnv("PAC7OV"))
 Call CopMFil(OM , WshEnv("PAC7OD") ,OM )
 Call DelFile (WshEnv("PAC7OD"))
 Call CopMFil(OM , WshEnv("PAC7OF") ,OM )
 Call DelFile (WshEnv("PAC7OF"))
 Call CopMFil(OM , WshEnv("PAC7OB") ,OM )
 Call DelFile (WshEnv("PAC7OB"))

 Set OM = FSO.GetFile(Rep_USR & "\GPRTOM.txt")
 FilFull = OM.size

If FilFull = 0 then  DelFile(OM) end if

End If

If BVPACAGP <> " " then
   Call Msg_Log (Array("1022" , BVPACAGP))
'-------------------------------------
Return = WshShell.Run(BVPACAGP , 1 , TRUE )
  Call Err_Cod(Return , 0 , BVPACAGP)
End If

If BVP_Updtpm = "YES" then
   Call Msg_Log (Array("1022" , "PCMPUF"))
'-------------------------------------
 WshEnv("PAC7AE") = Rep_SKEL & "\AE"
 WshEnv("SEMLOCK") = Rep_BASE & "\LO"
 WshEnv("SEMADMIN") = Rep_ABASE & "\LO"
 WshEnv("PAC7IC") = Rep_TMP & "\NUL.tmp"
'PAC7IC not used, on default
 WshEnv("PAC7QJ") = Rep_AJOURNAL & "\QJ"
 WshEnv("PAC7AJ") = Rep_JOURNAL & "\AJ"
 WshEnv("PAC7AN") = Rep_BASE & "\AN"
 WshEnv("PAC7AR") = Rep_BASE & "\AR"
 WshEnv("PAC7AY") = Rep_BASE & "\AY"

 WshEnv("PACGGN") = Rep_ABASE & "\AN"
 WshEnv("PACGGR") = Rep_ABASE & "\AR"
 WshEnv("PACGGU") = Rep_ABASE & "\GU"
 Call RunCmdLog ("BVPCMPUF")
 Call Err_Cod(Return , 0 , "PCMPUF")
end if
```

```
Call Msg_Log (Array("1024"))
'----------------------------
Call DeleteFldr(Rep_TMP)


Call Msg_Log (Array("1023"))
'----------------------------
Wscript.Quit (Return)

</script>
</job>
```

GPRT - Generated Files

Coding of GPRT output files created on disk

All output files generated by the GPRT procedure are created in the Temporary Files directory.

These files follow a special coding, so that the user can easily find his generated programs or reports.

Generated source and print files

These files are assigned the 'GPRT.' prefix.

For example :

Generated source	Print files
GPRTOB (Database Blocks)	GPRTIA (Report)
GPRTOQ (SQL)	
GPRTOD (Data)	GPRTID (Data)
GPRTOE (Screens - OSD)	GPRTIH (PEI)
GPRTOP (Programs)	GPRTIL (OSD Error Mes.)
GPRTOR (Reverse)	GPRTIN (PDM-Volumes)
GPRTOG (Client screens)	GPRTIK (OCS Error Mes.)
GPRTOV (Server screens)	GPRTII (ICS Generat. Err)
GPRTOF (e-Business)	

Error message files

These files are assigned the "ERR." prefix:

```
Input files : ERRLG and ERRLK
Input files : ERR.LG (OSD) and ERR.LK (OCS)
Output files: ERRGL and ERRGK
```

At the end of the procedure, a COPY order ensures the rotation from GL to LG and GK to LK.

On-line applications automatic revamping file

This file is assigned the "PAW." prefix:

PAW.GT contains the necessary elements for windowing.

Temporary files

There are files internal to the GPRT procedure.

These files are assigned the "WW" prefix and are deleted at the end of the procedure.

These files are assigned the "WX" prefix. They are deleted at the end of the procedure. If you want to retrieve them, you must change their location (copy in directory \USERS\...).

WXGI (VA Pac-GIP Interface)

WXGM (PAC700-type labels)

WXGN (Volumes on 265 characters).

Note concerning the generation of error messages

It is advisable to request the generation of Error Messages (GEO or GCO command) in batch mode rather than using the Generation & Print Commands screen.

The Batch Server, which processes the Generation-Print requests submitted from the 'GP' screen, does not perform the rotation of the generated sequential files; therefore there can be no cumulative generation.

As a result, error messages generated in prior on-line requests are lost.

In order to avoid this problem, the indexed Error Message file must be routinely loaded via the EMUP procedure after each sequential file generation.

By default, the GPRT procedure does not perform a cumulative generation of error messages, the LG and LK files being assigned as null files.

To activate the cumulative generation, assign the files as follows:

```
WshEnv("PAC7LG")  = RepT_USR & "\ERRLG.txt"
WshEnv("PAC7LK")  = RepT_USR & "\ERRLK.txt"
```

Processing the printouts in RTF format (GPRTG6.txt file)

The files generated in RTF format on the VA Pac server require to be converted into the ASCII character set before being processed by the VA Pac WorkStation.

Conversion command into ASCII character set:

bvptrans <source file> <destination file> ibm-923 ibm-850

These commands can be included in the BVPACAGP procedure.

EMLD - Loading of User-Defined Error Messages

EMLD - Introduction

The EMLD procedure performs the initial loading of user- defined error messages. These messages are obtained from the sequential output file of the GPRT procedure (GL-suffixed file).

Execution conditions

Prior execution of GPRT, with an error messages generation request.

Before the standard processing, perform an ASCII sort of the error messages file (PTUSGL).

EMLD - User Input

One '*' line with user code and password.

EMLD - Description of Steps

Sort of the generated sequential error messages: PTUSGL

Code	Physical Name	Type	Label
PAC7LG	User dir. : ERRGL	Input	Generated user error messages
PAC7GL	Tmp. dir. : ERRGL	Output	Sorted user error messages

Loading of user-defined error messages in an indexed file: PACL93

Code	Physical name	Type	Label
PAC7MB	User input	Input	Input Transactions
PAC7GL	Tmp dir. : ERRGL	Input	Sequential user-defined error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database Index file

Code	Physical name	Type	Label
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database Data file
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database Users
PAC7AR	Database dir. : AR	Input	Development Database Data file
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7EM	User dir. : ERRMSG	Output	User-defined indexed error messages file
PAC7IY	User dir. : EMLDIYL93	Report	Output reports
PAC7DD	User dir. : EMLDDDL93	Report	Authorization control

Return code :

- 8 : no authorization on batch procedure

EMLD - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'             - LOADING OF USER'S ERROR MESSAGES -
'
' ------------------------------------------------------
'
' THE EMLD PROCEDURE PERFORMS THE INITIAL LOADING OF USER
' DEFINED ERROR MESSAGES. THESE MESSAGES ARE OBTAINED
' FROM THE SEQUENTIAL OUTPUT FILE OF THE GPRT PROCEDURE
' (FILE WITH THE GL SUFFIX).
'
'
' INPUT :
'  -  USER IDENTIFICATION LINE (REQUIRED)
' ------------------------------------------------------
'
<job id=EMLD>

<script language="VBScript">
Dim MyProc
MyProc = "EMLD"
</script>

<script language="VBScript" src="INIT.vbs"/>

<script language="VBScript">
```

```
If c_error = 1 then Wscript.Quit (1) End If

Call BvpEnv("PTUSGL","PAC7LG",RepT_USR & "\ERRGL.txt")
If  FSO.FileExists(WshEnv("PAC7LG")) Then

Call Msg_Log (Array("1022" , "PTUSGL"))
'-----------------------------------
Call BvpEnv("PTUSGL","PAC7LG",RepT_USR & "\ERRGL.txt")
Call BvpEnv("PTUSGL","PAC7GL",Rep_TMP & "\ERRGL.txt")
Call RunCmdLog ("BVPTUSGL")

Call Msg_Log (Array("1022" , "PACL93"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7MB") = Fic_Input
Call BvpEnv("PACL93","PAC7GL",Rep_TMP & "\ERRGL.txt")
Call BvpEnv("PACL93","PAC7EM",RepT_USR & "\ERRMSG")
Call BvpEnv("PACL93","PAC7IY",Rep_USR & "\EMLDIYL93.txt")
Call BvpEnv("PACL93","PAC7DD",Rep_USR & "\EMLDDDL93.txt")
Call RunCmdLog ("BVPACL93")

If Return = 8 Then
Call Msg_Log (Array("1027"))
End If
Call Err_Cod(Return , 0 , "PACL93")

Else

Call Msg_Log (Array("1041" , RepT_USR & "\ERRGL.txt"))

End If

Call Msg_Log (Array("1023"))
'--------------------------------
Call DeleteFldr (Rep_TMP)
Wscript.Quit (Return)

</script>
</job>
```

EMUP - Update of User-Defined Error Messages

EMUP - Introduction

The EMUP procedure updates the User-Defined Error Message file.

These messages are obtained from the sequential file output by the GPRT procedure (GL-suffixed file) or from transactions for error message deletions at the entity level.

Execution conditions

The User-Defined Error Message file must exist.

In case of the creation and/or modification of error messages, the GPRT procedure must have been executed with the request for the generation of error messages.

Before the standard processing, perform an ASCII sort of the error messages file (PTUSGL).

EMUP - User Input

A '*' line per library containing entities whose error message(s) must be deleted:

Position	Length	Value	Meaning
2	1	'*'	Line code
3	8	uuuuuuuu	User code
11	8	pppppppp	User password
19	3	bbb	Library code

One command line per entity for which error message deletion is requested:

Position	Length	Value	Meaning
1	1	'D'	Transaction code (deletion)
2	2		Entity type; same as in CHOICE field
		'O '	Screen
		'D '	Data structure
		'S '	Segment
4	6		Entity code

EMUP - Description of Steps

Sort of the generated sequential error messages: PTUSGL

Code	Physical Name	Type	Label
PAC7LG	User dir. : ERRGL	Input	Generated user error messages
PAC7GL	Tmp. dir. : ERRGL	Output	Sorted user error messages

Update of indexed user-defined error messages: PACL92

Code	Physical name	Type	Label
PAC7GL	Tmp dir. : ERRGL	Input	Sequential user-defined error messages
PAC7AR	Database dir. : AR	Input	Development Database Data file
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database index
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database Data file
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database users
PAC7MB	User input	Input	Input transactions
PAC7EM	User dir. : ERRMSG	Output	User-defined error message indexed file
PAC7IU	User dir. : EMUPIUL92	Report	Transaction report
PAC7IX	User dir. : EMUPIXL92	Report	Error message report
PAC7DD	User dir. : EMUPDDL92	Report	Authorization option

Return code :

- 8 : no batch procedure authorization option.

EMUP - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'             - USER'S ERROR MESSAGES UPDATING -
'
' ------------------------------------------------------
'
' THE EMUP PROCEDURE UPDATES THE USER-DEFINED ERROR
' MESSAGE FILE. THESE MESSAGES ARE OBTAINED FROM THE
' SEQUENTIAL OUTPUT FILE OF THE GPRT PROCEDURE (FILE WITH
' A GL SUFFIX) OR FROM TRANSACTIONS FOR ERROR
' MESSAGE DELETIONS AT THE ENTITY LEVEL.
'
' INPUT :
'  -  USER IDENTIFICATION LINE (REQUIRED)
'  -  COMMAND LINE :
'  COL 1  : "D"    TRANSACTION CODE (DELETION)
'  COL 2  :  ENTITY TYPE; SAME AS IN CHOICE FIELD.
```

```
'            "O "   SCREEN
'            "D "   DATA STRUCTURE
'            "S "   SEGMENT
'  COL 4  : (6 CAR.) ENTITY CODE
' -----------------------------------------------------
'
<job id=EMUP>
<script language="VBScript">
MyProc = "EMUP"
Dim MyProc
</script>

<script language="VBScript" src="INIT.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If

Call BvpEnv("PTUSGL","PAC7LG",RepT_USR & "\ERRGL.txt")
If  FSO.FileExists(WshEnv("PAC7LG")) Then

Call Msg_Log (Array("1022" , "PTUSGL"))
'-----------------------------------
Call BvpEnv("PTUSGL","PAC7LG",RepT_USR & "\ERRGL.txt")
Call BvpEnv("PTUSGL","PAC7GL",Rep_TMP & "\ERRGL.txt")
Call RunCmdLog ("BVPTUSGL")

Call Err_Cod(Return , 0 , "PTUSGL")

Call Msg_Log (Array("1022" , "PACL92"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7MB") = Fic_Input
Call BvpEnv("PACL92","PAC7GL",Rep_TMP & "\ERRGL.txt")
Call BvpEnv("PACL92","PAC7EM",RepT_USR & "\ERRMSG")
Call BvpEnv("PACL92","PAC7IU",Rep_USR & "\EMUPIUL92.txt")
Call BvpEnv("PACL92","PAC7IX",Rep_USR & "\EMUPIXL92.txt")
Call BvpEnv("PACL92","PAC7DD",Rep_USR & "\EMUPDDL92.txt")
Call RunCmdLog ("BVPACL92")

If Return = 8 Then
Call Msg_Log (Array("1027"))
End If
Call Err_Cod(Return , 0 , "PACL92")

Else

Call Msg_Log (Array("1041" , RepT_USR & "\ERRGL.txt"))

End If
```

```
Call Msg_Log (Array("1023"))
'--------------------------------
Call DeleteFldr (Rep_TMP)
Wscript.Quit (Return)

</script>
</job>
```

PPAF - Generated Programs PAF Preprocessor

PPAF - Introduction

Using PAF operators, the PPAF procedure processes generated user programs containing SQL requests for access to the Database.

Execution conditions

None.

Implementation

This procedure may be executed in different ways:

- Either after the generation of programs via GPRT; its generated output is used as input to PPAF, before being compiled or stored in a source program library,
- Or by requesting the procedure in the command lines Before/After generated program; the appropriate JCL must have been previously entered in the selected options (PC screen).

PPAF - User Input

The input is the COBOL source code of programs containing PAF operators to be processed by the pre-processor before being compiled.

After the IDENTIFICATION DIVISION, each program contains a command line for the pre-processor. Its structure is as follows :

Position	Length	Value	Meaning
1	6	nnnnnn	COBOL line number
7	1	'*'	Comment
8	5	'TP '	On-line program OR
		'BATCH'	Batch program
14	5	'LIB:'	Fixed label
19	3	bbb	Library code
22	1	blank	Not used

Position	Length	Value	Meaning
23	5	nnnns	Session number - Session version
28	1	blank	Not used
29	2	--	Generation variant(s)
32	4	'AR:'	Fixed label
36	1	1	Database language code
38	4	'SC:'	Batch Language program skeleton
		'SG:'	On-line program skeleton
		'SR:'	COBOL program skeleton
42	1	1	Skeleton language
43	1	blank	Not used
44	6	'SINGLE'	Single quotes OR
		'DOUBLE'	Double quotes

Examples

000020*TP LIB: APP 2345 00 AR: F SG: F SINGLE

000020*BATCH LIB: APP 2300T 4 AR: F SC: F DOUBLE

This line is automatically generated by the GPRT procedure.

Printed output

This procedure prints an error report.

Result

The result of the PPAF procedure is the COBOL source in which PAF operators have been processed and calls to PAF batch or on-line sub-programs have been generated.

PPAF - Description of Steps

Preprocessor: PAFP10

Code	Physical name	Type	Label
PAC7AR	Database dir. : AR	Input	Development Database Data file
PAC7AN	Database dir. : AN	Input	Development Database Index file
PAC7AE	System - Skel dir. : AE	Input	Error messages

Code	Physical name	Type	Label
PAF80	User dir. : PAF80	Input	Generated programs
COB80	User dir. : COB80	Output	Generated programs to be compiled
PAFREP	User dir. : PPAFREP10 or PAFREP	Report	Error report

PPAF - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'         - VA PAC ACCESS FACILITY PRE-PROCESSING -
'
' ------------------------------------------------------
'
' USING PAF OPERATORS, THE PPAF PROCEDURE PROCESSES
' GENERATED USER PROGRAMS CONTAINING SQL REQUESTS FOR
' ACCESS TO THE DATABASE.
'
' USER INPUT IS THE COBOL SOURCE CODE OF PROGRAMS
' CONTAINING PAF OPERATORS TO BE PROCESSED BY
' BY THE PRE-PROCESSOR BEFORE COMPILATION.
'
' ------------------------------------------------------
'
<job id=PPAF>

<script language="VBScript">
Dim MyProc
MyProc = "PPAF"
</script>

<script language="VBScript" src="INIT.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If

Call Msg_Log (Array("1022" , "PAFP10"))
'-----------------------------------
WshEnv("COBSW") = "-N"
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
Call BvpEnv("PAFP10","PAF80",Rep_USR & "\PAF80.txt")
Call BvpEnv("PAFP10","COB80",Rep_USR & "\COB80.txt")
Call BvpEnv("PAFP10","PAFREP",Rep_USR & "\PPAFREP10.txt")
Call RunCmdLog ("BVPAFP10")
```

```
Call Err_Cod(Return , 0 , "PAFP10")

Call Msg_Log (Array("1023"))
'-----------------------------------
Call DeleteFldr (Rep_TMP)
Wscript.Quit (Return)


</script>
</job>
```

GPRC - COBOL API management

GPRC - Introduction

GPRT and the COBOL API: GPRC

This procedure makes it possible to use Client/Server services, such as Folders and Elementary components, in batch mode.

To do this, the GPRT procedure is completed by specific processing whose result is the GPRC procedure. It consists in the generation of sources for the COBOL API of the Folder manager. GPRC is a procedure dedicated to this type of generation ONLY.

For more information refer to the 'COBOL API User's Guide' manual.

GPRC - User Input

Refer to the description of GPRT user input.

GPRC - Description of Steps

Generation and Print: PACB

The generated source provided depends on the generation-print commands taken into account.

The entities which can use the COBOL API are:

- Programs,
- Macrostructures,
- Screens,
- Elementary Components.

For a complete information, refer to the GPRT description.

COBOL API extractor: PAPG1S

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages

Code	Physical name	Type	Label
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database Index file
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database Data file
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database Users
PAC7AN	Database dir. : AN	Input	Development Database Index file
PAC7AR	Database dir. : AR	Input	Development Database Data file
GENERE	Tmp dir.: GPRCOM	Input	GPRT generated source
PAC7W1	Tmp dir.: WW1	Output	Work file

COBOL API: PAPG5S

Code	Physical name	Type	Label
PAC7W1	Tmp dir. : WW1	Input	Input file
PAC7W2	Tmp dir. : WW2	Output	Output file

COBOL API generator: PAPG7S

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7EW	Tmp dir. : WEW	Output	Generation errors file
PAC7W2	Tmp dir. : WW2	Input	Intermediate file
PAC7W3	Tmp dir. : WW3	Input	Intermediate file
PAC7SA	System - Skel dir. : SA	Input	Skeleton of COBOL API labels

COBOL API - COBOL insertion: PAPG9S

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7EW	Tmp dir. : WEW	Input	Generation errors file
GENERE	Tmp dir. : GPRCOM	Input	GPRT generated source
PAC7W3	Tmp dir. : WW3	Output	Intermediate file
COB80	User dir. : GPRTOM	Input	Generated API COBOL source
PAC7ED	User dir. : GPRTED	Report	Errors report

GPRC - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'    - GENERATION AND PRINTING WITH API COBOL -
'
' ------------------------------------------------------
'
' IN ADDITION TO THE GENERATED ENTITIES, THE FILE MUST
' CONTAIN THE JCL REQUIRED TO COMPILE THEM,
' USING THE BEGINNING/END OF JCL JOB STREAM OPTIONS AND
' THE BEFORE/AFTER PROGRAM OPTIONS.
'
' ------------------------------------------------------
'
<job id=GPRC>

<script language="VBScript">
Dim MyProc
MyProc = "GPRC"
</script>
<script language="VBScript" src="INIT.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If


Dim CodLang
If base = "ADMIN"  Then
Call Msg_Log (Array("1028",base))
Wscript.Quit (0)
Else
 CodLang = WshShell.RegRead (Rep_SYS & "\GENLANG")
End If


 If Not FSO.FileExists( Rep_AJOURNAL & "\QJ") Then
 Call Msg_Log (Array("1022" , "PCMINI"))
'-------------------------------------
 WshEnv("PAC7QJ") =  Rep_AJOURNAL & "\QJ"
 Call RunCmdLog ("BVPCMINI")

 Call Err_Cod(Return , 0 , "PCMINI")
 End if

 If Not FSO.FileExists( Rep_ABASE & "\GK") Then
 Call Msg_Log (Array("1022" , "PTUIGK"))
'-------------------------------------
 WshEnv("PACGGK") =  Rep_ABASE & "\GK"
 Call RunCmdLog ("BVPTUIGK")

 Call Err_Cod(Return , 0 , "PTUIGK")
 End if
```

```
Call Msg_Log (Array("1022" , "PACB"))
'-----------------------------------
WshEnv("PAC7QJ") =  Rep_AJOURNAL & "\QJ"
WshEnv("PAC7AE") =  Rep_SKEL & "\AE"
WshEnv("PAC7AJ") = Rep_JOURNAL & "\AJ"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PAC7AY") = Rep_BASE & "\AY"
WshEnv("PAC7LB") = Rep_BASE & "\LB"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGK") = Rep_ABASE & "\GK"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7GS") = Rep_BASE & "\GS"
WshEnv("PAC7SC") = Rep_SKEL & "\SC" & CodLang
WshEnv("PAC7SF") = Rep_SKEL & "\SF" & CodLang
WshEnv("PAC7SG") = Rep_SKEL & "\SG" & CodLang
WshEnv("PAC7SN") = Rep_SKEL & "\SN" & CodLang
WshEnv("PAC7SR") = Rep_SKEL & "\SR" & CodLang
WshEnv("PAC7SS") = Rep_SKEL & "\SS" & CodLang
WshEnv("PAC7SL") = Rep_SKEL & "\SL" & CodLang
WshEnv("PAC7ME") = Fic_Input
Call BvpEnv("PACB","PAC7BM",Rep_TMP & "\WBM.tmp")
Call BvpEnv("PACB","PAC7DG",Rep_TMP & "\WDG.tmp")
Call BvpEnv("PACB","PAC7EB",Rep_TMP & "\WEB.tmp")
Call BvpEnv("PACB","PAC7EE",Rep_TMP & "\WEE.tmp")
Call BvpEnv("PACB","PAC7EG",Rep_TMP & "\WEG.tmp")
Call BvpEnv("PACB","PAC7EI",Rep_TMP & "\WEI.tmp")
Call BvpEnv("PACB","PAC7EN",Rep_TMP & "\WEN.tmp")
Call BvpEnv("PACB","PAC7EP",Rep_TMP & "\WEP.tmp")
Call BvpEnv("PACB","PAC7EQ",Rep_TMP & "\WEQ.tmp")
Call BvpEnv("PACB","PAC7ER",Rep_TMP & "\WER.tmp")
Call BvpEnv("PACB","PAC7EV",Rep_TMP & "\WEV.tmp")
Call BvpEnv("PACB","PAC7EW",Rep_TMP & "\WEW.tmp")
Call BvpEnv("PACB","PAC7OB",Rep_USR & "\GPRCOB.txt")
Call BvpEnv("PACB","PAC7GB",Rep_TMP & "\GPRCGB.tmp")
Call BvpEnv("PACB","PAC7OD",Rep_USR & "\GPRCOD.txt")
Call BvpEnv("PACB","PAC7GD",Rep_TMP & "\GPRCGD.tmp")
Call BvpEnv("PACB","PAC7OE",Rep_USR & "\GPRCOE.txt")
Call BvpEnv("PACB","PAC7GE",Rep_TMP & "\GPRCGE.tmp")
Call BvpEnv("PACB","PAC7OF",Rep_USR & "\GPRCOF.txt")
Call BvpEnv("PACB","PAC7GF",Rep_TMP & "\GPRCGF.tmp")
Call BvpEnv("PACB","PAC7OG",Rep_USR & "\GPRCOG.txt")
Call BvpEnv("PACB","PAC7GG",Rep_TMP & "\GPRCGG.tmp")
Call BvpEnv("PACB","PAC7GI",Rep_USR & "\GPRCGI.txt")

Call BvpEnv("PACB","PAC7GK",RepT_USR & "\ERRGK.txt")
Call BvpEnv("PACB","PAC7GL",RepT_USR & "\ERRGL.txt")

Call BvpEnv("PACB","PAC7GN",Rep_TMP & "\WXGN.tmp")
Call BvpEnv("PACB","PAC7GO",Rep_TMP & "\WGO.tmp")
Call BvpEnv("PACB","PAC7OP",Rep_USR & "\GPRCOP.txt")
Call BvpEnv("PACB","PAC7GP",Rep_TMP & "\GPRCGP.tmp")
```

```
Call BvpEnv("PACB","PAC7OQ",Rep_USR & "\GPRCOQ.txt")
Call BvpEnv("PACB","PAC7GQ",Rep_TMP & "\GPRCGQ.tmp")
Call BvpEnv("PACB","PAC7OR",Rep_USR & "\GPRCOR.txt")

Call BvpEnv("PACB","PAC7GR",Rep_TMP & "\GPRCGR.tmp")
Call BvpEnv("PACB","PAC7GT",Rep_USR & "\PAWGT.txt")
Call BvpEnv("PACB","PAC7OV",Rep_USR & "\GPRCOV.txt")
Call BvpEnv("PACB","PAC7GV",Rep_TMP & "\GPRCGV.tmp")
Call BvpEnv("PACB","PAC7G6",Rep_USR & "\GPRCG6.txt")
Call BvpEnv("PACB","PAC7DB",Rep_USR & "\GPRCDB.txt")
Call BvpEnv("PACB","PAC7IA",Rep_USR & "\GPRCIA.txt")
Call BvpEnv("PACB","PAC7ID",Rep_USR & "\GPRCID.txt")
Call BvpEnv("PACB","PAC7IK",Rep_USR & "\GPRCIK.txt")
Call BvpEnv("PACB","PAC7IL",Rep_USR & "\GPRCIL.txt")
Call BvpEnv("PACB","PAC7IM",Rep_USR & "\GPRCIM.txt")
Call BvpEnv("PACB","PAC7IN",Rep_USR & "\GPRCIN.txt")
Call BvpEnv("PACB","PAC7IW",Rep_USR & "\GPRCIW.txt")
Call BvpEnv("PACB","PAC7JG",Rep_TMP & "\WJG.tmp")
Call BvpEnv("PACB","PAC7KB",Rep_TMP & "\WKB.tmp")
Call BvpEnv("PACB","PAC7KD",Rep_TMP & "\WKD.tmp")
Call BvpEnv("PACB","PAC7KE",Rep_TMP & "\WKE.tmp")

Call BvpEnv("PACB","PAC7KF",Rep_TMP & "\WKF.tmp")
Call BvpEnv("PACB","PAC7KG",Rep_TMP & "\WKG.tmp")
Call BvpEnv("PACB","PAC7KM",Rep_TMP & "\WKM.tmp")
Call BvpEnv("PACB","PAC7KN",Rep_TMP & "\WKN.tmp")
Call BvpEnv("PACB","PAC7KP",Rep_TMP & "\WKP.tmp")
Call BvpEnv("PACB","PAC7KQ",Rep_TMP & "\WKQ.tmp")
Call BvpEnv("PACB","PAC7KR",Rep_TMP & "\WKR.tmp")
Call BvpEnv("PACB","PAC7KS",Rep_TMP & "\WKS.tmp")
Call BvpEnv("PACB","PAC7KU",Rep_TMP & "\WKU.tmp")
Call BvpEnv("PACB","PAC7KV",Rep_TMP & "\WKV.tmp")
Call BvpEnv("PACB","PAC7LG",Rep_TMP & "\NUL.tmp")
'PAC7LG not used, on default
Call BvpEnv("PACB","PAC7LI",Rep_TMP & "\WLI.tmp")
Call BvpEnv("PACB","PAC7LK",Rep_TMP & "\NUL.tmp")
'PAC7LK not used, on default
Call BvpEnv("PACB","PAC7MG",Rep_TMP & "\WMG.tmp")
Call BvpEnv("PACB","PAC7MV",Rep_TMP & "\WMV.tmp")
Call BvpEnv("PACB","PAC7SO",Rep_TMP & "\WSO.tmp")
Call BvpEnv("PACB","PAC7WA",Rep_TMP & "\WWA.tmp")
Call BvpEnv("PACB","PAC7W1",Rep_TMP & "\WW1.tmp")
Call BvpEnv("PACB","PAC7W2",Rep_TMP & "\WW2.tmp")
Call BvpEnv("PACB","PAC7W3",Rep_TMP & "\WW3.tmp")
Call BvpEnv("PACB","PAC7W4",Rep_TMP & "\WW4.tmp")
Call BvpEnv("PACB","PAC7W6",Rep_TMP & "\WW6.tmp")
Call BvpEnv("PACB","PAC7W7",Rep_TMP & "\WW7.tmp")
Call BvpEnv("PACB","PAC7W8",Rep_TMP & "\WW8.tmp")
Call BvpEnv("PACB","PAC7W9",Rep_TMP & "\WW9.tmp")
Call BvpEnv("PACB","SYSPAF",Rep_TMP & "\WPAF.tmp")
Call RunCmdLog ("BVPACB")

If Return < 10  then
Call Msg_Log (Array("1062"))
Return = 0
```

```
End if
If Return = 10  then
Call Msg_Log (Array("1063"))
Return = 0
End if

If Return > 10  then
Call Msg_Log (Array("1064"))
End if
Call Err_Cod(Return , 10 , "PACB")

 Call Msg_Log (Array("1022" , "COPY in OM"))
'-----------------------------------
If Not FSO.FileExists(Rep_TMP & "\GPRCOM.txt") Then
 Set LogGen = FSO.CreateTextFile(Rep_TMP & "\GPRCOM.txt", TRUE)
 LogGen.Close
End if
 OM = FSO.GetFile(Rep_TMP & "\GPRCOM.txt")

 Call CopMFil(OM , WshEnv("PAC7OP") ,OM )
 Call DelFile (WshEnv("PAC7OP"))
 Call CopMFil(OM , WshEnv("PAC7OQ") ,OM )
 Call DelFile (WshEnv("PAC7OQ"))
 Call CopMFil(OM , WshEnv("PAC7OE") ,OM )
 Call DelFile (WshEnv("PAC7OE"))
 Call CopMFil(OM , WshEnv("PAC7OR") ,OM )
 Call DelFile (WshEnv("PAC7OR"))
 Call CopMFil(OM , WshEnv("PAC7OG") ,OM )
 Call DelFile (WshEnv("PAC7OG"))
 Call CopMFil(OM , WshEnv("PAC7OV") ,OM )
 Call DelFile (WshEnv("PAC7OV"))
 Call CopMFil(OM , WshEnv("PAC7OD") ,OM )
 Call DelFile (WshEnv("PAC7OD"))
 Call CopMFil(OM , WshEnv("PAC7OF") ,OM )
 Call DelFile (WshEnv("PAC7OF"))
 Call CopMFil(OM , WshEnv("PAC7OB") ,OM )
 Call DelFile (WshEnv("PAC7OB"))

Call Msg_Log (Array("1022" , "PAPG1S"))
'-----------------------------------
WshEnv("PAC7AE") =  Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
Call BvpEnv("PACB","GENERE",Rep_TMP & "\GPRCOM.txt")
Call BvpEnv("PACB","PAC7W1",Rep_TMP & "\WW1.tmp")
Call RunCmdLog ("BVPAPG1S")
Call Err_Cod(Return , 0 , "PAPG1S")

Call Msg_Log (Array("1022" , "PAPG5S"))
'-----------------------------------
Call BvpEnv("PACB","PAC7W1",Rep_TMP & "\WW1.tmp")
Call BvpEnv("PACB","PAC7W2",Rep_TMP & "\WW2.tmp")
```

```
Call RunCmdLog ("BVPAPG5S")
Call Err_Cod(Return , 0 , "PAPG5S")

Call Msg_Log (Array("1022" , "PAPG7S"))
'-----------------------------------
WshEnv("PAC7AE") =  Rep_SKEL & "\AE"
Call BvpEnv("PACB","PAC7EW",Rep_TMP & "\WEW.tmp")
Call BvpEnv("PACB","PAC7W3",Rep_TMP & "\WW3.tmp")
Call BvpEnv("PACB","PAC7W2",Rep_TMP & "\WW2.tmp")
WshEnv("PAC7SA") = Rep_SKEL & "\SA" & CodLang
Call RunCmdLog ("BVPAPG7S")
Call Err_Cod(Return , 0 , "PAPG7S")

Call Msg_Log (Array("1022" , "PAPG9S"))
'-----------------------------------
WshEnv("PAC7AE") =  Rep_SKEL & "\AE"
Call BvpEnv("PACB","PAC7EW",Rep_TMP & "\WEW.tmp")
Call BvpEnv("PACB","PAC7W3",Rep_TMP & "\WW3.tmp")
Call BvpEnv("PACB","GENERE",Rep_TMP & "\GPRCOM.txt")
Call BvpEnv("PACB","COB80",Rep_USR & "\GPRCOM.txt")
Call BvpEnv("PACB","PAC7ED",Rep_USR & "\GPRCED.txt")
Call RunCmdLog ("BVPAPG9S")
Call Err_Cod(Return , 0 , "PAPG9S")


If BVPACAGP <> " " then
   Call Msg_Log (Array("1022" , BVPACAGP))
'-----------------------------------
Return = WshShell.Run(BVPACAGP , 1 , TRUE )
  Call Err_Cod(Return , 0 , BVPACAGP)
End If

If BVP_Updtpm = "YES" then
   Call Msg_Log (Array("1022" , "PCMPUF"))
'-------------------------------------
 WshEnv("PAC7AE") = Rep_SKEL & "\AE"
 WshEnv("SEMLOCK") = Rep_BASE & "\LO"
 WshEnv("SEMADMIN") = Rep_ABASE & "\LO"
 WshEnv("PAC7IC") = Rep_TMP & "\NUL.tmp"
'PAC7IC not used, on default
 WshEnv("PAC7QJ") = Rep_AJOURNAL & "\QJ"
 WshEnv("PAC7AJ") = Rep_JOURNAL & "\AJ"
 WshEnv("PAC7AN") = Rep_BASE & "\AN"
 WshEnv("PAC7AR") = Rep_BASE & "\AR"
 WshEnv("PAC7AY") = Rep_BASE & "\AY"
 WshEnv("PACGGN") = Rep_ABASE & "\AN"
 WshEnv("PACGGR") = Rep_ABASE & "\AR"
 WshEnv("PACGGU") = Rep_ABASE & "\GU"
 Call RunCmdLog ("BVPCMPUF")
 Call Err_Cod(Return , 0 , "PCMPUF")
end if


Call Msg_Log (Array("1024"))
'---------------------------
```

```
Call DeleteFldr(Rep_TMP)


Call Msg_Log (Array("1023"))
'----------------------------
Wscript.Quit (Return)

</script>
</job>
```

GPMC - Management of MCI Operator

GPMC - Introduction

Generation and Move Corresponding

The MCI operator entered in VA Pacbase is not interpreted by the PACB generator, but later by the two programs (UTIMCR and UTIMCI) which process the COBOL code output by the generator.

Basic rules:

The MOVE CORRESPONDING applies to two group fields: the first operand must be entered on the same line as the operator, and the second operand must be entered next to the first one or on a continuation line.

Each elementary element of the first group must have its equivalent in the second group to be included in the MOVE.

The comparison of the COBOL fields is based on their 'Data Element' code, i.e. the character string after the first dash (there is a prefix before this dash).

For example, in 'PREFIX-FIELD-NUMBER-ONE', the whole string 'FIELD-NUMBER-ONE' is searched for in the composition of the other group ; if it is found, it will be included in the MOVE CORRESPONDING.

When a group is followed by an index, the fields which are generated are generated in the same way.

Since none of the 'MOVE' statements generated in this way is controlled, the errors, if any, will be detected by the COBOL compiler.

The WKMCI file, written by UTIMCR, lists the lines of the MCI statements detected in the analyzed COBOL (one or two lines for each statement, depending on the user input) and is read by the UTIMCI program. The COB80 file which contains the generated COBOL is read by the two programs.

The final file (MCI80) is the image of the COB80 file ; in the original COBOL, it copies the MCI lines as comments followed by the induced MOVE statements.

GPMC - User Input

Refer to the description of GPRT user input.

GPMC - Description of Steps

Generation and Print: PACB

The provided generated source depends on the generation-print commands taken into account.

For more information, refer to the GPRT description.

MCI generator: UTIMCR

Code	Physical name	Type	Label
MCI80	User dir.: GPMCOM	Input	GPRT generated code
WKMCI	Tmp dir.: WWK	Output	Work file

Return codes:

- 4 : No MCI statement has been detected and the processing stops.

MCI generator: UTIMCI

Code	Physical name	Type	Label
MCI80	User dir.: GPMCOM	Input	GPRT generated code
WKMCI	Tmp dir.: WWK	Input	Work file
COB80	User dir.: COB80	Output	Result source

Return code:

- 8: Unknown Source or Target of an MCI operator

GPMC - Execution Script

```
' -------------------------------------------------------
'      VISUALAGE PACBASE
'
' -------------------------------------------------------
'            - MOVE CORRESPONDING GENERATION  -
'
' -------------------------------------------------------
'
' IN ADDITION TO THE GENERATED ENTITIES, THE FILE MUST
' CONTAIN THE JCL REQUIRED TO COMPILE THEM,
```

```
' USING THE BEGINNING/END OF JCL JOB STREAM OPTIONS AND
' THE BEFORE/AFTER PROGRAM OPTIONS.
'
' ------------------------------------------------------
'
<job id=GPMC>

<script language="VBScript">
Dim MyProc
MyProc = "GPMC"
</script>
<script language="VBScript" src="INIT.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If

Dim RetPacb
Dim CodLang
If base = "ADMIN"  Then
Call Msg_Log (Array("1028",base))
Wscript.Quit (0)
Else
 CodLang = WshShell.RegRead (Rep_SYS & "\GENLANG")
End If

 If Not FSO.FileExists( Rep_AJOURNAL & "\QJ") Then
 Call Msg_Log (Array("1022" , "PCMINI"))
'-----------------------------------
 WshEnv("PAC7QJ") =  Rep_AJOURNAL & "\QJ"
 Call RunCmdLog ("BVPCMINI")

 Call Err_Cod(Return , 0 , "PCMINI")
 End if

 If Not FSO.FileExists( Rep_ABASE & "\GK") Then
 Call Msg_Log (Array("1022" , "PTUIGK"))
'-----------------------------------
 WshEnv("PACGGK") =  Rep_ABASE & "\GK"
 Call RunCmdLog ("BVPTUIGK")

 Call Err_Cod(Return , 0 , "PTUIGK")
 End if

Call Msg_Log (Array("1022" , "PACB"))
'-----------------------------------
WshEnv("PAC7QJ") =  Rep_AJOURNAL & "\QJ"
WshEnv("PAC7AE") =  Rep_SKEL & "\AE"
WshEnv("PAC7AJ") = Rep_JOURNAL & "\AJ"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PAC7AY") = Rep_BASE & "\AY"
WshEnv("PAC7LB") = Rep_BASE & "\LB"
WshEnv("PACGGK") = Rep_ABASE & "\GK"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
```

```
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7GS") = Rep_BASE & "\GS"
WshEnv("PAC7SC") = Rep_SKEL & "\SC" & CodLang
WshEnv("PAC7SF") = Rep_SKEL & "\SF" & CodLang
WshEnv("PAC7SG") = Rep_SKEL & "\SG" & CodLang
WshEnv("PAC7SN") = Rep_SKEL & "\SN" & CodLang
WshEnv("PAC7SR") = Rep_SKEL & "\SR" & CodLang
WshEnv("PAC7SS") = Rep_SKEL & "\SS" & CodLang
WshEnv("PAC7SL") = Rep_SKEL & "\SL" & CodLang
WshEnv("PAC7ME") = Fic_Input
Call BvpEnv("PACB","PAC7BM",Rep_TMP & "\WBM.tmp")
Call BvpEnv("PACB","PAC7DG",Rep_TMP & "\WDG.tmp")
Call BvpEnv("PACB","PAC7EB",Rep_TMP & "\WEB.tmp")
Call BvpEnv("PACB","PAC7EE",Rep_TMP & "\WEE.tmp")
Call BvpEnv("PACB","PAC7EG",Rep_TMP & "\WEG.tmp")
Call BvpEnv("PACB","PAC7EI",Rep_TMP & "\WEI.tmp")
Call BvpEnv("PACB","PAC7EN",Rep_TMP & "\WEN.tmp")
Call BvpEnv("PACB","PAC7EP",Rep_TMP & "\WEP.tmp")
Call BvpEnv("PACB","PAC7EQ",Rep_TMP & "\WEQ.tmp")
Call BvpEnv("PACB","PAC7ER",Rep_TMP & "\WER.tmp")
Call BvpEnv("PACB","PAC7EV",Rep_TMP & "\WEV.tmp")
Call BvpEnv("PACB","PAC7EW",Rep_TMP & "\WEW.tmp")
Call BvpEnv("PACB","PAC7OB",Rep_USR & "\GPRTOB.txt")
Call BvpEnv("PACB","PAC7GB",Rep_TMP & "\GPRTGB.tmp")
Call BvpEnv("PACB","PAC7OD",Rep_USR & "\GPRTOD.txt")
Call BvpEnv("PACB","PAC7GD",Rep_TMP & "\GPRTGD.tmp")
Call BvpEnv("PACB","PAC7OE",Rep_USR & "\GPRTOE.txt")
Call BvpEnv("PACB","PAC7GE",Rep_TMP & "\GPRTGE.tmp")
Call BvpEnv("PACB","PAC7OF",Rep_USR & "\GPRTOF.txt")
Call BvpEnv("PACB","PAC7GF",Rep_TMP & "\GPRTGF.tmp")
Call BvpEnv("PACB","PAC7OG",Rep_USR & "\GPRTOG.txt")
Call BvpEnv("PACB","PAC7GG",Rep_TMP & "\GPRTGG.tmp")
Call BvpEnv("PACB","PAC7GI",Rep_USR & "\GPRTGI.txt")
Call BvpEnv("PACB","PAC7GK",RepT_USR & "\ERRGK.txt")
Call BvpEnv("PACB","PAC7GL",RepT_USR & "\ERRGL.txt")
Call BvpEnv("PACB","PAC7GM",RepT_USR & "\ERRGM.txt")
Call BvpEnv("PACB","PAC7GN",Rep_TMP & "\WXGN.tmp")
Call BvpEnv("PACB","PAC7GO",Rep_TMP & "\WGO.tmp")
Call BvpEnv("PACB","PAC7OP",Rep_USR & "\GPRTOP.txt")
Call BvpEnv("PACB","PAC7GP",Rep_TMP & "\GPRTGP.tmp")
Call BvpEnv("PACB","PAC7OQ",Rep_USR & "\GPRTOQ.txt")
Call BvpEnv("PACB","PAC7GQ",Rep_TMP & "\GPRTGQ.tmp")
Call BvpEnv("PACB","PAC7OR",Rep_USR & "\GPRTOR.txt")
Call BvpEnv("PACB","PAC7GR",Rep_TMP & "\GPRTGR.tmp")
Call BvpEnv("PACB","PAC7GT",Rep_USR & "\PAWGT.txt")
Call BvpEnv("PACB","PAC7OV",Rep_USR & "\GPRTOV.txt")
Call BvpEnv("PACB","PAC7GV",Rep_TMP & "\GPRTGV.tmp")
Call BvpEnv("PACB","PAC7G6",Rep_USR & "\GPRTG6.txt")
Call BvpEnv("PACB","PAC7DB",Rep_USR & "\GPRTDB.txt")
Call BvpEnv("PACB","PAC7IA",Rep_USR & "\GPRTIA.txt")
Call BvpEnv("PACB","PAC7ID",Rep_USR & "\GPRTID.txt")
Call BvpEnv("PACB","PAC7IK",Rep_USR & "\GPRTIK.txt")
Call BvpEnv("PACB","PAC7IL",Rep_USR & "\GPRTIL.txt")
Call BvpEnv("PACB","PAC7IM",Rep_USR & "\GPRTIM.txt")
```

```
Call BvpEnv("PACB","PAC7IN",Rep_USR & "\GPRTIN.txt")
Call BvpEnv("PACB","PAC7IO",Rep_USR & "\GPRTIO.txt")
Call BvpEnv("PACB","PAC7IW",Rep_USR & "\GPRTIW.txt")
Call BvpEnv("PACB","PAC7JG",Rep_TMP & "\WJG.tmp")
Call BvpEnv("PACB","PAC7KB",Rep_TMP & "\WKB.tmp")
Call BvpEnv("PACB","PAC7KD",Rep_TMP & "\WKD.tmp")
Call BvpEnv("PACB","PAC7KE",Rep_TMP & "\WKE.tmp")
Call BvpEnv("PACB","PAC7KF",Rep_TMP & "\WKF.tmp")
Call BvpEnv("PACB","PAC7KG",Rep_TMP & "\WKG.tmp")
Call BvpEnv("PACB","PAC7KM",Rep_TMP & "\WKM.tmp")
Call BvpEnv("PACB","PAC7KN",Rep_TMP & "\WKN.tmp")
Call BvpEnv("PACB","PAC7KP",Rep_TMP & "\WKP.tmp")
Call BvpEnv("PACB","PAC7KQ",Rep_TMP & "\WKQ.tmp")
Call BvpEnv("PACB","PAC7KR",Rep_TMP & "\WKR.tmp")
Call BvpEnv("PACB","PAC7KS",Rep_TMP & "\WKS.tmp")
Call BvpEnv("PACB","PAC7KU",Rep_TMP & "\WKU.tmp")
Call BvpEnv("PACB","PAC7KV",Rep_TMP & "\WKV.tmp")
Call BvpEnv("PACB","PAC7LG",Rep_TMP & "\NUL.tmp")
'PAC7LG not used, on default
Call BvpEnv("PACB","PAC7LI",Rep_TMP & "\WLI.tmp")
Call BvpEnv("PACB","PAC7LK",Rep_TMP & "\NUL.tmp")
'PAC7LK not used, on default
Call BvpEnv("PACB","PAC7LM",Rep_TMP & "\NUL.tmp")
'PAC7LM not used, on default
Call BvpEnv("PACB","PAC7MG",Rep_TMP & "\WMG.tmp")
Call BvpEnv("PACB","PAC7MV",Rep_TMP & "\WMV.tmp")
Call BvpEnv("PACB","PAC7SO",Rep_TMP & "\WSO.tmp")
Call BvpEnv("PACB","PAC7WA",Rep_TMP & "\WWA.tmp")
Call BvpEnv("PACB","PAC7W1",Rep_TMP & "\WW1.tmp")
Call BvpEnv("PACB","PAC7W2",Rep_TMP & "\WW2.tmp")
Call BvpEnv("PACB","PAC7W3",Rep_TMP & "\WW3.tmp")
Call BvpEnv("PACB","PAC7W4",Rep_TMP & "\WW4.tmp")
Call BvpEnv("PACB","PAC7W6",Rep_TMP & "\WW6.tmp")
Call BvpEnv("PACB","PAC7W7",Rep_TMP & "\WW7.tmp")
Call BvpEnv("PACB","PAC7W8",Rep_TMP & "\WW8.tmp")
Call BvpEnv("PACB","PAC7W9",Rep_TMP & "\WW9.tmp")
Call BvpEnv("PACB","SYSPAF",Rep_TMP & "\WPAF.tmp")
Call RunCmdLog ("BVPACB")
RetPacb = Return

If Return < 10  then
Call Msg_Log (Array("1062"))
Return = 0
End if

If Return = 10  then
Call Msg_Log (Array("1063"))
Return = 0
End if

If Return > 10  then
Call Msg_Log (Array("1064"))
End if

Call Err_Cod(Return , 10 , "PACB")
```

```
If RetPacb < 10 Then
 Call Msg_Log (Array("1022" , "COPY in OM"))
'-----------------------------------
If Not FSO.FileExists(Rep_USR & "\GPRTOM.txt") Then
 Set LogGen = FSO.CreateTextFile(Rep_USR & "\GPRTOM.txt", TRUE)
 LogGen.Close
End if
OM = FSO.GetFile(Rep_USR & "\GPRTOM.txt")

 Call CopMFil(OM , WshEnv("PAC7OP") ,OM )
 Call DelFile (WshEnv("PAC7OP"))
 Call CopMFil(OM , WshEnv("PAC7OQ") ,OM )
 Call DelFile (WshEnv("PAC7OQ"))
 Call CopMFil(OM , WshEnv("PAC7OE") ,OM )
 Call DelFile (WshEnv("PAC7OE"))
 Call CopMFil(OM , WshEnv("PAC7OR") ,OM )
 Call DelFile (WshEnv("PAC7OR"))
 Call CopMFil(OM , WshEnv("PAC7OG") ,OM )
 Call DelFile (WshEnv("PAC7OG"))
 Call CopMFil(OM , WshEnv("PAC7OV") ,OM )
 Call DelFile (WshEnv("PAC7OV"))
 Call CopMFil(OM , WshEnv("PAC7OD") ,OM )
 Call DelFile (WshEnv("PAC7OD"))
 Call CopMFil(OM , WshEnv("PAC7OF") ,OM )
 Call DelFile (WshEnv("PAC7OF"))
 Call CopMFil(OM , WshEnv("PAC7OB") ,OM )
 Call DelFile (WshEnv("PAC7OB"))

 Set OM = FSO.GetFile(Rep_USR & "\GPRTOM.txt")
 FilFull = OM.size

If FilFull = 0 then  DelFile(OM) end if

Call Msg_Log (Array("1022" , "UTIMCR"))
'-----------------------------------
Call BvpEnv("UTIMCR","MCI80",Rep_USR & "\GPRTOM.txt")
Call BvpEnv("UTIMCR","WKMCI",Rep_TMP & "\WWK.tmp")
Call RunCmdLog ("BVPUTMCR")

Call Err_Cod(Return , 8 , "UTIMCR")

If Return = 8 Then
'No MCI found
'-------------
Call Msg_Log (Array("1067"))
end if

If Return = 4 Then

   Ret = fso.MoveFile (Rep_USR & "\GPRTOM.txt" , _
   Rep_TMP & "\GPRTOM.txt")
   If Ret <> 0 Then
      Call Msg_Log (Array("1025", "MOVE-OM", Ret))
      Wscript.Quit (0)
```

```
   End If

Call Msg_Log (Array("1022" , "UTIMCI"))
'-----------------------------------
Call BvpEnv("UTIMCI","COB80",Rep_USR & "\GPRTOM.txt")
Call BvpEnv("UTIMCI","MCI80",Rep_TMP & "\GPRTOM.txt")
Call BvpEnv("UTIMCI","WKMCI",Rep_TMP & "\WWK.tmp")
Call RunCmdLog ("BVPUTMCI")

Call Err_Cod(Return , 0 , "UTIMCI")
End If
End If

If BVPACAGP <> " " then
   Call Msg_Log (Array("1022" , BVPACAGP))
'-------------------------------------
Return = WshShell.Run(BVPACAGP , 1 , TRUE )
  Call Err_Cod(Return , 0 , BVPACAGP)
End If

If BVP_Updtpm = "YES" then
   Call Msg_Log (Array("1022" , "PCMPUF"))
'-----------------------------------
 WshEnv("PAC7AE") = Rep_SKEL & "\AE"
 WshEnv("SEMLOCK") = Rep_BASE & "\LO"
 WshEnv("SEMADMIN") = Rep_ABASE & "\LO"
 WshEnv("PAC7IC") = Rep_TMP & "\NUL.tmp"
'PAC7IC not used, on default
 WshEnv("PAC7QJ") = Rep_AJOURNAL & "\QJ"
 WshEnv("PAC7AJ") = Rep_JOURNAL & "\AJ"
 WshEnv("PAC7AN") = Rep_BASE & "\AN"
 WshEnv("PAC7AR") = Rep_BASE & "\AR"
 WshEnv("PAC7AY") = Rep_BASE & "\AY"
 WshEnv("PACGGN") = Rep_ABASE & "\AN"
 WshEnv("PACGGR") = Rep_ABASE & "\AR"
 WshEnv("PACGGU") = Rep_ABASE & "\GU"
 Call RunCmdLog ("BVPCMPUF")
 Call Err_Cod(Return , 0 , "PCMPUF")
end if
Call Msg_Log (Array("1024"))
'----------------------------
Call DeleteFldr(Rep_TMP)


Call Msg_Log (Array("1023"))
'----------------------------
Wscript.Quit (Return)

</script>
</job>
```

Chapter 3. Extractions

PACX - Introduction

The extraction procedure allows to perform various types of data extractions from the Development Database via a PAF extractor (selection of criteria).

See chapter 'UPDP - Update from PAF Tables' in 'The Developer's Procedures' manual.

Data is extracted as transactions that can be used as input to the following procedures:

- UPDT
- UPDP
- CPSN (If the optional 'Partitioned Database Manager' utility is available.)

Execution conditions

None since the Database is not directly updated by this procedure.

PACX - User Input Common to all Extractors

Position	Length	Value	Meaning
2	1	'*'	Line code
3	8	uuuuuuuu	User code
11	8	pppppppp	Password
19	3	bbb	Extraction library code, or target Library code if RMEN with upload
22	4	nnnn	Session number (blank=current ses.)
26	1	'T'	Session status if Test session
29	4	cccc	Extractor code (1)
33	1	'1'	Formatting for UPDT
		'2'	CPSN : formatting for UPDT with explicit transaction codes
		' '	No formatting for UPDT
34	1	'1'	Formatting for UPDP (PAF)
		'2'	CPSN : formatting for UPDP with explicit transaction codes

Position	Length	Value	Meaning
		' '	No formatting for UPDP (PAF)
35	1	'1'	Formatting for CPSN
		' '	No formatting for CPSN
40	3	ppp	DSMS Product Code
43	6	nnnnnn	DSMS Change number (DSMS Function only)
49	1		Lock processing
		' '	Lock extraction with user code ('*'-line user code)
		'1'	No lock extraction
		'2'	Lock extraction with user code (original user code)
		'N'	For RMEN only : no extraction of locked entities by an other user
50	1	' '	No transfer of password
		'1'	Password transfer
69	3	bbb	Library code for the '*'-line of the output file(s) (For EXTR, EXLI and EXUE only)
76	5	nnnnT	Session number for the '*'-line of the output file(s) (For EXTR, EXLI and EXUE only)

(1) The possible values for the extractor code are:

- EXTR: Extraction of entities (extracted transactions are sorted).
- EXTA: Extraction of entities (extracted transactions are sorted, according to the input identification lines order. So if each request is preceded by a '*' line, extracted transactions will be sorted in the order of the requests). The formatting is forced to UPDT.
- EXUE: Extraction of user entities
- EXLI: Extraction of libraries or library sub-networks (formatting for UPDP, UPDT or CPSN).
- EXPJ: Extraction of Journal (formatting for CPSN is not possible)
- EXPU: Extraction for purge (formatting for CPSN is not possible)
- RMEN: Extraction of entities for upload/replacement/ recoding (formatting for CPSN is not possible). RMEN is subject to a separate purchase agreement.
- CPSN: comparison of sub-networks or entities.

Important

- One extractor type only for each run: If the procedure detects more than one type of extractors, it will take only the first one into account.
- The formatting type of the first '*' line only is taken into account.
- Formatting for CPSN: This procedure is part of the 'Partitioned Database Manager' optional utility. Its use is therefore subject to a separate purchase agreement.
- Maximum number of input '*' lines : 1 for RMEN and EXPJ, 1000 for EXTR, EXTA, EXUE and EXPU.
- The Pacdesign/Pacbench technical locks are not extracted by PACX.

Results

The PACX procedure produces:

- A report which contains the list of executed programs and the number of generated transactions.
- A list of requests with possible associated errors.
- One or more execution reports depending on the type of extractor.

EXTR/EXTA - Extraction of Entities

EXTR/EXTA - Introduction

These extractor types allow the selection of all or only part of an entity.

If the request has an 'ALL' type, the whole entity is extracted, i.e. the entity itself but also all the entities it uses, as well as entities used by those, and so on. Used entities that are not cross-referenced are not extracted.

Depending on the type of formatting requested, the resulting file can be used as input to the UPDT, UPDP or CPSN procedures (if the request is of the 'ALL', 'ONLY' or 'EXPT' type; the formatting for CPSN is not allowed).

For EXTA, the formatting is forced to UPDT.

EXTR/EXTA - User Input

One or two command lines per entity to be extracted.

First line :

Position	Length	Value	Meaning
2	1	'W'	Line code
3	1	'1'	Line number
4	2	'EX'	

Position	Length	Value	Meaning
6	1		Library selection code:
		'U'	Library alone
		'C'	Library and its upper-level libraries
		'+'	Library and its upper-level libraries with identification lines ('*' lines) generation
7	33	Choice	Entity to be extracted, coded in the same way as the 'Choice' field in TP.
40	4		Extraction type (1):
		' '	Entity alone (required for EXTA)
		'ALL '	Entity and used entities
		'ONLY'	Entity and only used entities whose types are specified in the following part of the line
		'EXPT'	Entity and used entities, except those whose types are specified in the following part of the line
44			12-position table (3 char./position) containing exceptions or selections
			'DEL': Element
			'DBD': Database Block
			'DST': Data Structure
			'SEG': Segment
			'RPT': Report
			'TXT': Text
			'VOL': PDM Volume
			'PGM': Program
			'DLG': Dialog
			'SCR': Screen
			'PIA': Parameterized Input Aid
			'MET': Methodology
			'CME': Client Meta-Entity
			'CLR': Client User Relation
			'$tt': User Entity (tt = Meta-entity type)
			'EME': Extension Meta-Entity
			'ERL : Extension User Relation
			'Ytt': Extension User Entity (tt = Meta-Entity type (1))

(1) The use of the 'ALL', 'ONLY' and 'EXPT' values is submitted to the acquisition of the 'In-Depth extractor' optional module.

Second line (continuation line for selections and exceptions):

Position	Length	Value	Meaning
2	1	'W'	Line code
3	1	'2'	Line number
44			12-position table (3 characters per position) containing the exceptions or selections

(1) The Meta-Entity type are :

CE	Elementary Component
CS	eBusiness Application
C1	SCM tool interface
DO	Folder
D1	Publishing Documents
D2	Document Type Definition
F1	External Files
G1	Generation-print command
MC	Communication Monitor
MS	Message
OP	Operation
PT	Part
SB	SOAP Binding
SI	Initialization Server
SV	Service
UM	UML Interface
VL	Logical View
5Q	Quality rule definition
7E	Extraction master path

The EXTR procedure also works with choices that are specific to the Development Database.

These choices must be entered from the seventh position, in the following way:

//A__CCCXXXXXX

where A is the methodology code and CCC the entity local code.

Type of extraction

- The 'in-depth extractor' option ('ALL', 'EXPT' or 'ONLY' extraction type) is not available for EXTA. For this procedure, the value must be blank.
- By default, the extraction of a Data Structure extracts its Segments. To prevent the Segments from being extracted, you must enter 'EXPTSEG' as the extraction type. This is possible, even if the 'in-depth extractor' option is not available.
- The extraction of a Dialog extracts only the Dialog by default. To extract the Dialog 's screens, enter 'ALL'.
- Same as above for the extraction of a Meta-Entity and its User Entities.
- The extraction stops at the first level of selection or exception.

  Example: Extraction of a Program with 'EXPTSEG' - The Elements used by the Segments used by the Program are not extracted since the extractor does not consider those segments.

Printed output

The procedure produces a list of extracted entities:

- Sorted for EXTR,
- In the order of the requests for EXTA.

EXUE - Extraction of User Entities Contents

EXUE - Introduction

The EXUE procedure extracts the contents of User Entities according to the Meta-Entity type code, formatted as simple records in a sequential file.

The EXUE procedure is part of the Dictionary Extensibility Function which is an optional component and whose use is subject to a separate purchase agreement.

See the 'Dictionary Extensibility' Manual.

EXUE - User Input

One command line per user entity:

Position	Length	Value	Meaning
2	4	'W1EX'	Line code
6	1	'$'	Client UE Extraction identifier

Position	Length	Value	Meaning
		'Y'	Extension UE Extraction identifier
7	1		Library selection code:
		'U'	Selected Library
		'C'	Selected Library + higher level Libr.
8	2	CC	Meta-Entity call type

Printed output

The EXUE procedure prints a list of the extracted UEs.

Result

The output of the EXUE procedure is a sequential file with a fixed format in which the contents of the selected user entities are recorded.

The length of each record is 230 characters.

Each record includes:

- A common part containing all the characteristics necessary to identify each extracted line.
- A specific part whose format depends on the meta-entity description.

PACX - Description of Steps

Extraction: PACX

This step extracts transactions according to user input.

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7AN	Database dir. : AN	Input	Development Database index
PAC7AR	Database dir. : AR	Input	Development Database data
PAC7AY	Database dir. : AY	Input	Development Database extension data
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database index
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database data

Code	Physical name	Type	Label
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database users
PAC7PJ	Save dir.: PJ	Input	Archived transactions
PAC7MB	User input	Input	User input
PAC7MA	NUL	Input	CPSN Master file
PAC7ES	NUL	Input	CPSN Slave file
PAC7BM	Tmp dir.: WBM	Input/Output	User input
PAC7MM	Tmp dir.: WMM	Input/Output	EXPU Work file
PAC7MJ	Tmp dir.: WMJ	Input/Output	EXPJ Work file
PAC7TE	Tmp dir.: WTE	Input/Output	RMEN Work file
PAC7RE	Tmp dir.: WRE	Input/Output	RMEN Work file
PAC7RM	Tmp dir.: WRM	Input/Output	RMEN Work file
PAC7WD	Tmp dir.: WWD	Input/Output	Extracted transactions
SYSEXT	Tmp dir.: WSY	Input/Output	Indexed Work File
PAC7RQ	NUL	Input/Output	Work file
PAC7MV	User dir.: PACXMV	Output	Extracted transactions for UPDT
PAC7MR	User dir.: PACXMR	Output	Extracted transactions for REOR (EXPU)
PAC7MX	User dir.: PACXMX	Output	Non extracted entities (PACX)
PAC7GY	User dir.: PACXGY	Output	Extracted transactions for UPDP
PAC7TD	User dir.: PACXTD	Output	Extracted transactions for CPSN
PAC7UE	User dir.: PACXUE	Output	Extracted transactions for EXUE
PAC7IA	User dir.: PACXIA	Report	General printout of the program stream
PAC7DD	User dir.: PACXDD	Report	Errors on input transactions
PAC7ED	User dir.: PACXED	Report	Extractions report
PAC7EE	User dir.: PACXEE	Report	Extractions report
PAC7EG	User dir.: PACXEG	Report	Extractions report
PAC7EM	User dir.: PACXEM	Report	Extractions report
PAC7EP	User dir.: PACXEP	Report	Extractions report
PAC7EQ	User dir.: PACXEQ	Report	Extractions report

Code	Physical name	Type	Label
PAC7EU	User dir.: PACXEU	Report	Extractions report
PAC7EZ	User dir.: PACXEZ	Report	Extractions report

Return codes:

- 0: No error
- 4: Error in user input (detailed in PAC7EE) or in EXTR/EXUE extractions (detailed in PAC7EZ)
- 8: Error in '*' line (detailed in PAC7DD) or in EXLI (Database not available)

PACX - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'           - EXTRACTIONS FROM DATABASE -
'           - EXTRACTIONS COMPARATOR    -
' ------------------------------------------------------
'
' THE PACX PROCEDURE ALLOWS TO PERFORM VARIOUS TYPES
' OF DATA EXTRACTIONS FROM THE DEVELOPMENT DATABASE
' VIA PAF EXTRACTOR.
'
' POSSIBLE VALUES FOR THE EXTRACTOR CODE INCLUDE:
' - EXTR:  EXTRACTION OF ENTITIES
' - EXTA:  EXTRACTION OF ENTITIES (EXTRACTED TRANSACTIONS
'          ARE SORTED, ACCORDING TO THE INPUT
'          IDENTIFICATION LINES ORDER.
'          EACH REQUEST IS THUS PRECEDED BY A "*" LINE,
'          EXTRACTED TRANSACTIONS WILL BE SORTED IN THE
'          REQUEST ORDER).
' - EXUE:  EXTRACTION OF USER ENTITIES
' FOLLOWING VALUES ARE RESERVED FOR THE ADMINISTRATOR:
' - EXLI:EXTRACTION OF LIBRARIES OR LIBRARY SUB-NETWORKS
' - EXPJ:EXTRACTION OF JOURNAL (FORMATTING FOR CPSN IS
'        NOT POSSIBLE)
' - EXPU:EXTRACTION OF ENTITIES TO BE PURGED
'        (FORMATTING FOR CPSN IS NOT POSSIBLE)
' - RMEN:EXTRACTION OF ENTITIES FOR UPLOAD/REPLACEMENT/
'        RECODING (FORMATTING FOR CPSN IS NOT POSSIBLE).
'        RMEN IS SUBJECT TO A SEPARATE PURCHASE AGREEMENT
' - CPSN:COMPARISON OF SUB-NETWORKS.
'
' ------------------------------------------------------
'
<job id=PACX>

<script language="VBScript">
Dim MyProc
```

```
MyProc = "PACX"
</script>

<script language="VBScript" src="INIT.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If

Call Msg_Log (Array("1022" , "PACX"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PAC7AY") = Rep_BASE & "\AY"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7PJ") = Rep_SAVE & "\PJ"
WshEnv("PAC7MB") = Fic_Input
Call BvpEnv("PACX","PAC7BM",Rep_TMP & "\WBM.tmp")
Call BvpEnv("PACX","PAC7WD",Rep_TMP & "\WWD.tmp")
Call BvpEnv("PACX","PAC7MM",Rep_TMP & "\WMM.tmp")
Call BvpEnv("PACX","PAC7MJ",Rep_TMP & "\WMJ.tmp")
Call BvpEnv("PACX","PAC7TE",Rep_TMP & "\WTE.tmp")
Call BvpEnv("PACX","PAC7RE",Rep_TMP & "\WRE.tmp")
Call BvpEnv("PACX","PAC7RM",Rep_TMP & "\WRM.tmp")
Call BvpEnv("PACX","PAC7RQ",Rep_TMP & "\NUL.tmp")

Call BvpEnv("PACX","PAC7MA",Rep_TMP & "\NUL.tmp")
'PAC7MA not used, on default
Call BvpEnv("PACX","PAC7ES",Rep_TMP & "\NUL.tmp")
'PAC7ES not used, on default

'Example of Output File reuse in next procedure :
' Call BvpEnv("PACX","PAC7xx",RepT_USR & "\PACXxx.txt")
'With  RepT_USR is Global User Directory.

'One for each procedure : Rep_USR & "\PACXxx.txt"
'One for all the procedure : RepT_USR & "\PACXxx.txt"

'Call BvpEnv("PACX","PAC7UE",Rep_USR & "\PACXUE.txt")
Call BvpEnv("PACX","PAC7UE",RepT_USR & "\PACXUE.txt")

'Call BvpEnv("PACX","PAC7GY",Rep_USR & "\PACXGY.txt")
Call BvpEnv("PACX","PAC7GY",RepT_USR & "\PACXGY.txt")

'Call BvpEnv("PACX","PAC7TD",Rep_USR & "\PACXTD.txt")
Call BvpEnv("PACX","PAC7TD",RepT_USR & "\PACXTD.txt")

'Call BvpEnv("PACX","PAC7MV",Rep_USR & "\PACXMV.txt")
Call BvpEnv("PACX","PAC7MV",RepT_USR & "\PACXMV.txt")

'Call BvpEnv("PACX","PAC7MR",Rep_USR & "\PACXMR.txt")
Call BvpEnv("PACX","PAC7MR",RepT_USR & "\PACXMR.txt")
```

```
'Call BvpEnv("PACX","PAC7MX",Rep_USR & "\PACXMX.txt")
Call BvpEnv("PACX","PAC7MX",RepT_USR & "\PACXMX.txt")

Call BvpEnv("PACX","PAC7IA",Rep_USR & "\PACXIA.txt")
Call BvpEnv("PACX","PAC7DD",Rep_USR & "\PACXDD.txt")
Call BvpEnv("PACX","PAC7ED",Rep_USR & "\PACXED.txt")
Call BvpEnv("PACX","PAC7EE",Rep_USR & "\PACXEE.txt")
Call BvpEnv("PACX","PAC7EG",Rep_USR & "\PACXEG.txt")
Call BvpEnv("PACX","PAC7EM",Rep_USR & "\PACXEM.txt")
Call BvpEnv("PACX","PAC7EP",Rep_USR & "\PACXEP.txt")
Call BvpEnv("PACX","PAC7EQ",Rep_USR & "\PACXEQ.txt")
Call BvpEnv("PACX","PAC7EU",Rep_USR & "\PACXEU.txt")
Call BvpEnv("PACX","PAC7EZ",Rep_USR & "\PACXEZ.txt")

Call BvpEnv("PACX","SYSEXT",Rep_TMP & "\WSY.tmp")
Call RunCmdLog ("BVPACX")
If Return = 4 Then
Call Msg_Log (Array("1030"))
End If
If Return = 8 Then
Call Msg_Log (Array("1057"))
End If
Call Err_Cod(Return , 0 , "PACX")

Call Msg_Log (Array("1024"))
'-----------------------------------
Call DeleteFldr (Rep_TMP)

Call Msg_Log (Array("1023"))
'-----------------------------------
Wscript.Quit (Return)

</script>
</job>
```

Chapter 4. Personalized Extraction/Automated Documentation

Foreword

The PAF+/Extraction and the PDM+/Outline functions can be used separately or together.

PAF+ is used to write the Extraction Master Path and execute it when the PTEx is a User Extractor.

PDM+ is used to write and execute the Master Outline (PTEd).

The PAF-PDM functions are used when the Master outline calls an Extraction Master Path of the Macro-Command type.

- If you use the PAF+/Extraction function alone, you can generate User Extractor programs and possibly format the extracted data.
- If you use the PDM+/Outline function alone, you can create skeletons to standardize the printing of Volumes (standard Print Options, Text instances always called, standardized calls).
- If you use both functions together, PAF+ extracts data from the Database. This data is processed by PDM+ and finally printed in a Volume.

For more information on these functions, refer to the 'Pacbase Access Facility (PAF)' and the 'Personalized Documentation Manager (PDM)' manuals.

Personalized Extractions - PAF+

XPAF - Validation of an Extraction Master Path

XPAF - Introduction

The Extraction Master Path validation procedure, XPAF, is used to perform specific extractions that the standard procedures cannot perform. See the 'Pacbase Access Facility (PAF)' reference manual.

Results

The type of result depends on whether or not the extracted domain is to be integrated into a Document : Macro-Command or User Extraction program.

A Macro-Command is a subroutine to be activated in a GPRT print request (choice: PCV).

A User Extraction program is a Source Program to be compiled and executed.

Prerequisite

In order to use this procedure, the Database Manager must update the Database with the transaction file, supplied for installation, which contains the .PPTEX extension Meta-Entity, whose type is 7E (VINS procedure).

The GS file, initialized by the LDGS procedure, must pre-exist.

Implementation

Before the procedure can be executed, the user must define an instance of this extension meta-entity (Y7E). Its Definition and Description determine the characteristics and format of the general extraction program.

Abnormal execution

Whatever the cause of the abend, the procedure can be re-executed once the problem has been solved.

Printed output

This procedure prints a validation report and a simulation of the Extraction Master Path.

XPAF - User Input

One '*' line per library and session to be consulted

Position	Length	Value	Meaning
2	1	'*'	Line code
3	8	uuuuuuuu	User code
11	8	pppppppp	User password
19	3	bbb	Library code
22	4	nnnn	Session number
26	1	'T'	Session version
68	1	' '	Standard print
		'1'	Uppercase print

One command line 'EX' for the following elements:

Position	Length	Value	Meaning
2	2	'EX'	Line code
4	2		Extension ME Type (Y7E by default)
6	6	eeeeee	User Entity code
			Warning: Specify library and session if the MEs whose instances are to be extracted are in a parallel sub-network (UEs extractions managed by the WorkStation for example)
12	3	bbb	Library code
15	4	nnnn	Session number
19	1	'T'	Session version
20	6	'UPDATE'	Update of GS
		SPACE	Check of the presence of the Master Path in GS (no update). Check of the user entity's use in the sub-network.

Examples

*user passwordLIB

EX7EEXT001_____UPDATE

*user passwordLIB

EX7EEXT002

XPAF - Description of Steps

Access and validation: PTEX30

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database Index
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database Data file
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database Users

Code	Physical name	Type	Label
PAC7AN	Database dir. : AN	Input	Development Database Index file
PAC7AR	Database dir. : AR	Input	Development Database Data file
PAC7AY	Database dir. : AY	Input	Development Database Extension Data
PAC7MB	User input	Input	User Input
PAC7SP	System - Skel dir. : SP	Input	Variable skeleton file
PAC7GS	Database dir. : GS	Input / Output	Extraction Paths
PAC7ED	Tmp dir. : WED	Output	Report passed on to printing program
PAC7GP	Tmp dir. : WGP	Output	Temporary generated source
PAC7DD	User dir. : XPAFDDX30	Report	Report

Extractor generation: PTEX80

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error Messages
PAC7SF	System - Skel dir. : SF	Input	Fixed skeleton file
PAC7GP	Tmp dir. : WGP	Input	Source file generated by PTEX30
PAC7ST	User dir. : PAF80	Output	Generated source to be translated

Preprocessor: PAFP10

Code	Physical name	Type	Label
PAC7AR	Database dir. : AR	Input	Development Database Data file
PAC7AN	Database dir. : AN	Input	Development Database Index file
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAF80	User dir. : PAF80	Input	Generated programs
COB80	User dir. : COB80	Output	Generated programs to be compiled
PAFREP	User dir. : PPAFREP10 or PAFREP	Report	Error report

PTEX printing: PTEXD0

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database Index file
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database Data file
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database Users
PACGGY	Admin Database - Base dir. : AY	Input	Administration Database Extension
PAC7ED	Tmp. dir. : WED	Input	PTEX30 Report
PAC7GS	Base dir. : GS	Input/Output	Extraction Paths
PAC7RD	User dir. : XPAFRDXD0	Report	Control report

XPAF - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'                 - XPAF (PAF EXTENSION) -
'
' ------------------------------------------------------
'
' THE EXTRACTION MASTER PATH VALIDATION PROCEDURE,
' XPAF, ALLOWS FOR THE SIMULATION OF SPECIFIC EXTRACTIONS
' THAT THE STANDARD PROCEDURES ARE NOT ABLE TO PERFORM.
'
' INPUT :
'  -  USER IDENTIFICATION LINE (REQUIRED)
'     COL  2 : "*"
'     COL  3 : USERIDXX
'     COL 11 : PASSWORD
'     COL 19 : (BBB)    LIBRARY CODE
'     COL 22 : (4 N)    SESSION NUMBER
'     COL 26 : (1 CAR.) SESSION VERSION
'     COL 68 : " "      STANDARD PRINT
'              "1"      UPPERCASE PRINT
'  -  COMMAND LINE :
'  COL  2  : "EX"       LINE CODE
'  COL  4  :  (2 CAR.)  METAENTITY  TYPE (7E BY DEFAULT)
'  COL  6  :  (6 CAR.)  USER ENTITY CODE
'  COL 12  :  (BBB)     LIBRARY CODE     (IF THE U.E.O.
```

```
'  COL 15  :  (4 N)     SESSION NUMBER    ARE IN PARALLEL
'  COL 19  :  (1 CAR.)  SESSION VERSION   SUB-NETWORK)
'  COL 20  :  "UPDATE"  UPDATE OF GS
'             "      "  CHECK OF THE PRESENCE OF THE
'                       MASTER PATH IN GS.
'
' ------------------------------------------------------
'
<job id=XPAF>

<script language="VBScript">
Dim MyProc
MyProc = "XPAF"
</script>

<script language="VBScript" src="INIT.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If

Call Msg_Log (Array("1022" , "PTEX30"))
'-------------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PAC7AY") = Rep_BASE & "\AY"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
Call BvpEnv("PTEX30","PAC7ED",Rep_TMP & "\WED.tmp")
Call BvpEnv("PTEX30","PAC7DD",Rep_USR & "\XPAFDDX30.txt")
Call BvpEnv("PTEX30","PAC7GP",Rep_TMP & "\WGP.tmp")
WshEnv("PAC7GS") = Rep_BASE & "\GS"
WshEnv("PAC7MB") = Fic_Input
WshEnv("PAC7SP") = Rep_SKEL & "\SP"
Call RunCmdLog ("BVPTEX30")

If Return <> 8 Then
Call Err_Cod(Return , 0 , "PTEX30")
End If

If Return = 0 Then

Call Msg_Log (Array("1022" , "PTEX80"))
'-------------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
Call BvpEnv("PTEX80","PAC7GP",Rep_TMP & "\WGP.tmp")
WshEnv("PAC7SF") = Rep_SKEL & "\SF"
Call BvpEnv("PTEX80","PAC7ST",Rep_USR & "\PAF80.txt")
Call RunCmdLog ("BVPTEX80")
Call Err_Cod(Return , 0 , "PTEX80")

Call Msg_Log (Array("1022" , "PAFP10"))
'-------------------------------------
```

```
WshEnv("COBSW") = "-N"
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
Call BvpEnv("PAFP10","PAF80",Rep_USR & "\PAF80.txt")
Call BvpEnv("PAFP10","COB80",Rep_USR & "\COB80.txt")
Call BvpEnv("PAFP10","PAFREP",Rep_USR & "\PAFREP.txt")
Call RunCmdLog ("BVPAFP10")
Call Err_Cod(Return , 0 , "PAFP10")

End If

Call Msg_Log (Array("1022" , "PTEXD0"))
'-------------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PACGGY") = Rep_ABASE & "\AY"
Call BvpEnv("PTEXD0","PAC7ED",Rep_TMP & "\WED.tmp")
WshEnv("PAC7GS") = Rep_BASE & "\GS"
Call BvpEnv("PTEXD0","PAC7RD",Rep_USR & "\XPAFRDXD0.txt")
Call RunCmdLog ("BVPTEXD0")

Call Err_Cod(Return , 0 , "PTEXD0")

Call Msg_Log (Array("1024"))
'-----------------------------------
Call DeleteFldr (Rep_TMP)

Call Msg_Log (Array("1023"))
'-----------------------------------
Wscript.Quit (Return)


</script>
</job>
```

XPAF - Operation of an Extraction Master Path

Execution of a user extractor (E-type PTEx)

Once validated, compiled, and linked, a User Extractor is ready for execution.

Execution of a macro-command (M-type PTEx):

Once validated, compiled, and linked, a Macro-Command is not ready for execution. It must be called in a Master Outline.

Note

An Extraction Master Path is independent of the Database in which it is defined and described.

XPDM - Validation of a Master Outline

XPDM - Introduction

A Master Outline is a P-type Volume ('V' entity) designed to be called in another PDM Volume. Its functions are to:

- Memorize general descriptions (print option, for example) so that they do not have to be redefined in each Volume.
- Print the information extracted via an Extraction Master Path. This function may be recursive.

If no serious error is detected, the XPDM procedure updates the Extraction Master Path file (GS). It can also be used without updating the GS file.

See the 'Personalized Documentation Manager' manual for more details.

Abnormal execution

Whatever the cause of the abend, the procedure can be re-executed once the problem has been solved.

Printed output

This procedure prints the description of a Master Outline, as well as the comments, and a list of the anomalies found, if any.

XPDM - User Input

One '*' line to define the context.

Position	Length	Value	Meaning
2	1	'*'	Line code
3	8	uuuuuuuu	User code
11	8	pppppppp	User password
19	3	bbb	Library code
22	4	nnnn	Session number
26	1	'T'	Session version
68	1	' '	Standard print
		'1'	Uppercase print

One 'EP' line for the following elements :

Position	Length	Value	Meaning
2	2	'EP'	Line code
4	6	vvvvvv	Volume code
10	6	'UPDATE'	GS file update
		SPACE	Check of the volume's presence in GS Check of the volume's use in the sub-network. No GS file update if presence or use.

Examples

*user passwordLIB

EPMANUALUPDATE

*user passwordLIB

EPMANUAL

XPDM - Description of Steps

Extraction of master outline: PTED30

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database Index file
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database Data file
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database Users
PAC7AN	Database dir. : AN	Input	Development Database Index file
PAC7AR	Database dir. : AR	Input	Development Database Data file
PAC7AY	Database dir. : AY	Input	Development Database Extension data
PAC7MB	User input	Input	User Input
PAC7GS	Database dir. : GS	Input/Output	Extraction Paths
PAC7ED	Tmp dir. : WED	Output	File for report

Code	Physical name	Type	Label
PAC7SG	Tmp dir. : WSG	Output	GS-Update preparation
PAC7DD	User dir. : XPDMDDD30	Report	Report

GS update and printing of the master outline: PTED60

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database Index file
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database Data file
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database Users
PACGGY	Admin Database - Base dir. : AY	Input	Administration Database Extension
PAC7ED	Tmp dir. : WED	Input	Print file
PAC7SG	Tmp dir.: WSG	Input	GS Update preparation file
PAC7GS	Database dir. : GS	Output	Extraction Paths
ETATGP	User dir. : XPDMGPD60	Report	Output report

XPDM - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'                 - XPDM   (PDM EXTENSION) -
'
' ------------------------------------------------------
'
' A MASTER OUTLINE IS A P-TYPE VOLUME ("V" ENTITY)
' DESIGNED TO BE CALLED IN ANOTHER PDM VOLUME.
'
' INPUT :
'  -  USER IDENTIFICATION LINE (REQUIRED)
'     COL  2 : "*"
'     COL  3 : USERIDXX
'     COL 11 : PASSWORD
'     COL 19 : (BBB)    LIBRARY CODE
'     COL 22 : (4 N)    SESSION NUMBER
'     COL 26 : (1 CAR.) SESSION VERSION
```

```
'     COL 68 : " "      STANDARD PRINT
'              "1"      UPPERCASE PRINT
'  -  COMMAND LINE :
'  COL  2  : "EP"       LINE CODE
'  COL  4  :  (6 CAR.)  REPORT CODE
'  COL 10  :  "UPDATE"  UPDATE OF GS
'             "      "  CHECK OF THE PRESENCE OF VOLUME
'                       IN GS.
'
' ------------------------------------------------------
'
<job id=XPDM>

<script language="VBScript">
Dim MyProc
MyProc = "XPDM"
</script>

<script language="VBScript" src="INIT.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If

Call Msg_Log (Array("1022" , "PTED30"))
'-------------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PAC7AY") = Rep_BASE & "\AY"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
Call BvpEnv("PTED30","PAC7DD",Rep_USR & "\XPDMDDD30.txt")
Call BvpEnv("PTED30","PAC7ED",Rep_TMP & "\WED.tmp")
WshEnv("PAC7GS") = Rep_BASE & "\GS"
WshEnv("PAC7MB") = Fic_Input
Call BvpEnv("PTED30","PAC7SG",Rep_TMP & "\WSG.tmp")
Call RunCmdLog ("BVPTED30")
Call Err_Cod(Return , 0 , "PTED30")

Call Msg_Log (Array("1022" , "PTED60"))
'-------------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PACGGY") = Rep_ABASE & "\AY"
Call BvpEnv("PTED60","PAC7ED",Rep_TMP & "\WED.tmp")
WshEnv("PAC7GS") = Rep_BASE & "\GS"
Call BvpEnv("PTED60","PAC7SG",Rep_TMP & "\WSG.tmp")
Call BvpEnv("PTED60","ETATGP",Rep_USR & "\XPDMGPD60.txt")
Call RunCmdLog ("BVPTED60")
Call Err_Cod(Return , 0 , "PTED60")
```

```
Call Msg_Log (Array("1024"))
'--------------------------------
Call DeleteFldr (Rep_TMP)

Call Msg_Log (Array("1023"))
'-----------------------------------
Wscript.Quit (Return)


</script>
</job>
```

Extraction Master Path and Outline File

PRGS - Printing of Master Path / Outline File

PRGS - Introduction

The PRGS procedure prints the content of the PAC7GS file, where the Master Outlines and Extraction Master Paths are stored.

Result

A printout showing the Extraction Master Path and the associated Master Outlines.

PRGS - User Input

One '*' line to identify the user.

Position	Length	Value	Meaning
2	1	'*'	Line code
3	8	uuuuuuuu	User code
11	8	pppppppp	User password

PRGS - Description of Steps

Printing of the master path and outline file: PTEP90

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Database dir. : AN	Input	Administration Database Index file

Code	Physical name	Type	Label
PACGGR	Admin Database - Database dir. : AR	Input	Administration Database Data file
PACGGU	Admin Database - Database dir. : GU	Input	Administration Database Users
PAC7GS	Database dir. : GS	Input	Extraction Paths
PAC7MB	User input	Input	User Input
PAC7DD	User dir. : PRGSDDP90	Report	Output Report
ETATGS	User dir. : PRGSGSP90	Report	Master Path and Outline file report

PRGS - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'           -  PRINT OF MASTER PATH FILE -
'
' ------------------------------------------------------
'
' THE PRGS PROCEDURE PRINTS THE CONTENTS OF THE
' PAC7GS FILE, WHERE MASTER OUTLINES AND EXTRACTION
' MASTER PATHS ARE STORED.
'
' ------------------------------------------------------
'
<job id=PRGS>

<script language="VBScript">
Dim MyProc
MyProc = "PRGS"
</script>

<script language="VBScript" src="INIT.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If

Call Msg_Log (Array("1022" , "PTEP90"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
Call BvpEnv("PTEP90","PAC7DD",Rep_USR & "\PRGSDDP90.txt")
WshEnv("PAC7GS") = Rep_BASE & "\GS"
Call BvpEnv("PTEP90","ETATGS",Rep_USR & "\PRGSGSP90.txt")
```

```
WshEnv("PAC7MB") = Fic_Input
Call RunCmdLog ("BVPTEP90")
If Return = 8 Then
Call Msg_Log (Array("1027"))
End If
Call Err_Cod(Return , 0 , "PTEP90")


Call Msg_Log (Array("1023"))
'-----------------------------------
Call DeleteFldr (Rep_TMP)
Wscript.Quit (Return)


</script>
</job>
```

Chapter 5. Batch Update

UPDP - Update from PAF Tables

UPDP - Introduction

The UPDP procedure performs an update of the Database from a sequential file which is the image of PAF tables.

The operating principle of UPDP is very similar to that of UPDT, with the exception that input transactions have a different format.

Abnormal execution

Refer to the 'Abnormal Execution' section of the UPDT procedure.

UPDP - User Input / Update Rules / Results

The sequential file of input transactions is produced by a PAF extractor program or by the PACX procedure. Its records reflect the PAF tables format. For a detailed description of these tables, see the documentation of the Pacbase Access Facility Tables.

Position	Length	Meaning
1	1	Transaction code (C, M, X, A or D, B, S)
2	10	PAF table code
12	299	PAF table contents (as described in the PAF Tables Reference Manual)

There are restrictions on the Client and Extension User Entities Definition and Description tables.

The size of the UPDP input file is 310 characters long while the size of these tables exceeds 310 characters. The records must then be re-formatted in the following manner:

Client and Extension User Entities Definition Tables - $TTDEF and YTTDEF.

Position	Length	Meaning
1	1	Transaction code (C, M, X, A or D, B, S)
2	10	Table code

Position	Length	Meaning
12	1	Record continuation code: blank character for the first record, any character for the continuation records.
13	1	Separator (1)
14	55	Explicit keywords
69	305	Contents of the concatenated columns

Client and Extension User Entities Description tables - $TTDxx and YTTDxx.

Position	Length	Meaning
1	1	Transaction code (C, M, X, A or D, B)
2	10	Table code
12	1	Record continuation code: blank character for the first record, any character for the continuation records
13	1	Separator (1)
14	30	User Entity code
44	305	Contents of the concatenated columns

(1) If the separator is set to blank, data items are concatenated one after the other ; the length of each data item is respected.

If the separator is specified, data items are separated with this character. The non-significant blanks located at the end of the data item are optional.

The separator must not be present in the data item.

Update rules

Update transactions are not sorted.

Each set of transactions impacting a library or session must be preceded by an ASSIGN table code line.

Position	Length	Value	Meaning
2	10	'ASSIGN'	Table code
12	8	uuuuuuuu	User code
20	8	pppppppp	Password
28	3	bbb	Library code
31	4	ssss	Session number

Position	Length	Value	Meaning
		' '	current session
35	1	'T'	Session status: Test session
40	3	ppp	Product code (in case of a Database under DSMS control)
43	6	nnnnnn	Product number (in case of a Database under DSMS control)
49	1		Top generated by the extractor
		' '	Manual transactions
		'N'	Extractor transactions
		'G'	Transactions output from the retrieval of PG25 or PP25

When the update is performed while the on line mode is active, the input transaction flow must be preceded by a CHECKP table code line.

(Refer to the description of the UPDT output.)

Position	Length	Value	Meaning
2	10	'CHECKP'	Table code
12	4	nnnn	Number of transactions processed between two pauses or checkpoints
16	4	'UPDT'	Update procedure
20	2	nn	LAN Platforms: Pause time, in seconds, between two update sets

Printed output

Refer to the description of the UPDT output.

Result

Refer to the description of the UPDT result.

UPDP - Description of Steps

Database consistency check: PTUBAS

Code		Type	Label
PAC7AR	Base dir.: AR	Input	Development Database data

Code		Type	Label
PAC7AE	System - Skel dir.: AE	Input	Error messages
PAC7DS		Report	Validity report

.Return code:

This utility sends a return code 4 and causes an ABEND if the Database is inconsistent.

Transaction formatting: PAF900

Code	Physical name	Type	Label
PAC7AR	Database dir.: AR	Input	Development Database Data file
PAC7AN	Database dir.: AN	Input	Development Database Index file
PAC7AE	System - Skel. dir.: AE	Input	Error messages
PACGGR	Admin. Database - Base dir.: AR	Input	Administration Database Data file
PACGGN	Admin. Database - Base dir.: AN	Input	Administration Database Index file
PACGGU	Admin. Database - Base dir.: GU	Input	Administration Database Users
PAC7GY	User input	Input	Update transactions
PAC7MV	Tmp dir.: WMV	Output	Formatted transactions (must be able to contain all input transactions as well as elementary deletion transactions generated by multiple deletion transactions) (length=170)
PAC7ME	Tmp dir.: WME	Output	Work file (length=372)
PAC7MW	Tmp dir.: WMW	Output	Work file (length=170)
PAC7MX	Tmp dir.: WMX	Output	Work file (length=743)
PAC7MY	Tmp dir.: WMY	Output	Work file (length=743)

Update of the Development Database: PACA15

Code	Physical name	Type	Label
PAC7AR	Base dir. : AR	Output	Development Database Data file

Code	Physical name	Type	Label
PAC7AN	Base dir. : AN	Output	Development Database Index file
PAC7AY	Base dir. : AY	Output	Development Database Extension
PAC7AJ	Journal dir. : AJ	Output	Development Database Journal
PAC7AE	System - Skel. dir. : AE	Input	Error messages
PACGGN	Admin. Base - Base dir. : AN	Input	Administration Database Index file
PACGGR	Admin. Base - Base dir. : AR	Input	Administration Database Data file
PACGGY	Admin. Base - Base dir. : AY	Input	Administration Database Extension
PACGGU	Admin. Base - Base dir. : GU	Input	Administration Database Users
PAC7DC	Base dir. : DC	Input	Development Database elements DSMS file
PAC7ME	Tmp dir. : WME	Input	Work file
PAC7MV	Tmp dir. : WMV	Input	Update transactions
PAC7RB	User dir. : UPDPRBA15	Output	UPDT erroneous transactions (length=80)
PAC7RY	User dir. : UPDPRYA15	Output	UPDP erroneous transactions (length=310)
PAC7IE	User dir. : UPDPIEA15	Report	Update report (length=132)
PAC7IF	User dir. : UPDPIFA15	Report	List of erroneous transactions (length=132)

The list of transactions specific to a user is preceded by a banner with this user's code.

Return codes :

- 0 : OK without error
- 2 : Warning
- 4 : Error

UPDP - Execution Script

```
' -----------------------------------------------------
'      VISUALAGE PACBASE
'
' -----------------------------------------------------
'            - BATCH UPDATE FROM PAF TABLES -
```

```
'
' ------------------------------------------------------
'
' THE UPDP PROCEDURE PERFORMS AN UPDATE OF THE DATABASE
' FROM A SEQUENTIAL FILE REFLECTING PAF TABLES.
'
' THE SEQUENTIAL FILE OF INPUT TRANSACTIONS IS PRODUCED
' BY A PAF EXTRACTOR PROGRAM. ITS RECORDS MIRROR
' THE PAF TABLES.
' EACH SET OF TRANSACTIONS IMPACTING A LIBRARY OR SESSION
' MUST BE PRECEDED BY AN ASSIGN TABLE CODE LINE.
' WHEN THE UPDATE IS PERFORMED WHILE THE TP IS ACTIVE
' (ON PLATFORMS THAT SUPPORT THIS FUNCTIONALITY),
' THE INPUT TRANSACTION FLOW MUST BE PRECEDED BY A CHECKP
' TABLE CODE LINE.
' ------------------------------------------------------
'
<job id=UPDP>

<script language="VBScript">
Dim MyProc
MyProc = "UPDP"

</script>

<script language="VBScript" src="INIT.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If

Call Msg_Log (Array("1022" , "PTUBAS"))
'-------------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
Call BvpEnv("PTUBAS","PAC7DS",Rep_USR & "\UPDTDSBAS.txt")
Call RunCmdLog ("BVPTUBAS")
WshVolEnv("RC") = Return
If Return = 4 Then
Call Msg_Log (Array("1051"))
End If
Call Err_Cod(Return , 0 , "PTUBAS")

'Input File extracted from PACX
'in RepT_USR is Global User Directory.


Call Msg_Log (Array("1022" , "PAF900"))
'-------------------------------------
WshEnv("PAC7GY") = Fic_Input
If Not FSO.FileExists(WshEnv("PAC7GY")) Then
  Call Msg_Log (Array("1004" , "PAC7GY"))
  Msg = Nls_Lib
  EndJob (1)
  Wscript.Quit (1)
```

```
End If

WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
Call BvpEnv("PAF900","PAC7ME",Rep_TMP & "\WME.tmp")
Call BvpEnv("PAF900","PAC7MW",Rep_TMP & "\WMW.tmp")
Call BvpEnv("PAF900","PAC7MV",Rep_TMP & "\WMV.tmp")
Call BvpEnv("PAF900","PAC7MX",Rep_TMP & "\WMX.tmp")
Call BvpEnv("PAF900","PAC7MY",Rep_TMP & "\WMY.tmp")
Call RunCmdLog ("BVPAF900")
Call Err_Cod(Return , 0 , "PAF900")

Call Msg_Log (Array("1022" , "PACA15"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AJ") = Rep_JOURNAL & "\AJ"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PAC7AY") = Rep_BASE & "\AY"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGY") = Rep_ABASE & "\AY"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
Call BvpEnv("PACA15","PAC7DC",Rep_BASE & "\DC")
Call BvpEnv("PACA15","PAC7IE",Rep_USR & "\UPDPIEA15.txt")
Call BvpEnv("PACA15","PAC7IF",Rep_USR & "\UPDPIFA15.txt")
WshEnv("SEMLOCK") = Rep_BASE & "\LO"
WshEnv("SEMADMIN") = Rep_ABASE & "\LO"
Call BvpEnv("PACA15","PAC7ME",Rep_TMP & "\WME.tmp")
Call BvpEnv("PACA15","PAC7MV",Rep_TMP & "\WMV.tmp")
Call BvpEnv("PACA15","PAC7RB",Rep_USR & "\UPDPRBA15.txt")
Call BvpEnv("PACA15","PAC7RY",Rep_USR & "\UPDPRYA15.txt")
Call RunCmdLog ("BVPACA15")
If Return = 2 Then
Call Msg_Log (Array("1061"))
End If
If Return = 4 Then
Call Msg_Log (Array("1060"))
End If
Call Err_Cod(Return , 4 , "PACA15")

Call Msg_Log (Array("1024"))
'-----------------------------------
Call DeleteFldr (Rep_TMP)

Call Msg_Log (Array("1023"))
'-----------------------------------
Wscript.Quit (Return)
```

```
</script>
</job>
```

UPDT - Update

UPDT - Introduction

The Database update procedure (UPDT) executes a batch update of the Database. It allows access to all the libraries according to the authorizations of the different users.

With the DSMS facility (DSM), this procedure reads the VisualAge Pacbase elements file (DC).

Execution conditions

Since the Database is updated, the AR, AN, AJ and AY files must be closed to on-line use, except for hardware environments that support concurrent on-line and batch access.

Note

For very large updates (in terms of number of transactions, about 5000), before executing this procedure, it may be necessary to:

- Save, archive and restore the Database to increase the space allocated to the files or to physically reorganize the files in order to make all the free space initially provided available.
- Temporarily suppress Journalization

  (See chapter 'Database Management Utilities', subchapter 'Database Restoration', in 'The Administrator's Procedures' manual').

This procedure updates the current session number in two cases:

- When it is the first connection of the day to the Database, and
- When it contains a Database Freeze request submitted by the Administrator (see 'The Administrator's Procedures' manual)

Abnormal execution

Refer to chapter 'Overview', subchapter 'Abnormal Endings' in 'the Administrator's Procedures' manual.

There are two types of abnormal executions:

- Abnormal execution which occurs before the execution of the BVPACA15 program, or during the opening of files in this program. The procedure can be restarted after the problem is corrected.
- Abnormal execution which occurs during the execution of the BVPACA15 program. The Database is left in an inconsistent state if there is no rollback. If the problem appeared during input-output on a Database file, the printed error message and the file status will lead you to the solution.

  If the procedure execution has stopped with the error 'Short label already used', the Database remains consistent and the procedure can be restarted after the label has been corrected.

In either case, you can re-start the procedure only by using a backup file and apply the archived transactions subsequent to this backup (REST procedure).

UPDT - User Input / Update Rules / Results

Update rules

Each set of transactions impacting a Library must be preceded by a *-type line which specifies the context.

These transactions are not sorted.

Printed output

The two printed output generated by this procedure are:

- A global report on the update,
- A list of the rejected update transactions.

They are printed by the user, and the transaction groups are separated by a flag.

Result

Output of the UPDT procedure is:

- A Database ready to be used in on-line or batch mode.
- A Journal file of the transactions that have modified the Database (as long as there was no inhibit request during the last restoration).

  These transactions are made up of a common part which contains the action code, a line identifier and a specific part which is detailed in the following sections for each Description of entity.

Action codes

Action code	Label
C	Creation of a line in the library
M	Modification of a line.
Blank	Creation or modification of a line, depending on its presence or absence in the library.
X	Creation or modification with possible use of ampersand (&).
D	Deletion of one line.
B	Multiple lines deletion, starting with this line.
R	End of multiple lines deletion up to and including this line.
S	Complete deletion of an entity

Note about deletions

If an entity is used in several Libraries, deletions in a lower Library are rejected.

It is possible to globally delete (using ACTION CODE 'B') an entity and all of its uses in Screens, Reports or Segments. However, these deletions will be effective only in update Libraries.

The B code generates elementary deletion transactions.

The S code can be used on an entity definition only, one transaction only will be journalized. Checks will be done before the update.

Caution

A field which is not valued is not modified. Enter the '&' character to reset this field to blank.

Specific action codes: 'F' and 'P'

The 'F' and 'P' action codes are used in extractions for updates.

The 'F' value is used to force an update, i.e. after an extraction (via EXLI or any other extractor), it allows the creation of an incomplete Definition so that the X-references to these entities (usually User Entities) can be satisfied, a sort being impossible.

This code triggers the update of the Database.

The 'P' value allows an identification line to be assigned to all the Description lines that follow without updating the Definition of this entity (e.g. 'P' lines of a Program in a Library where the Definition exists only in a higher Library).

Checkpoints

This specification enables you to request synchronization points during the UPDT batch update.

You determine the frequency of the checkpoints (ex: a frequency equal to 0100 means that a checkpoint will be carried out after every 100 processed transactions).

Frequency of checkpoints during a batch update

For the UPDT batch update, you determine the frequency of checkpoints via a 'Y'-type line located before the first '*' line of the update flow. This line must have the following format:

Pos.	Len.	Value	Meaning
2	1	'Y'	Line code
4	4	nnnn	Frequency of checkpoints (default value: 0000)
8	2	nn	Pause time at each checkpoint

For the REST or RESY restoration, you determine the frequency of checkpoints via the user input defined for these procedures.

Concurrent batch-online update

The use of checkpoints in the BVPACA15 program of the UPDT procedure makes it possible to run this procedure concurrently with the on-line mode. This UPDT-online concurrency must be reserved to exceptional small transaction sets.

Actually the execution of the UPDT procedure during the online session may cause stoppages between 2 successive points, which can cause an increase of online response times.

In the case of a non-fatal abort (if the journal is full or if there is a problem on the call of a checkpoint), you can start the procedure again after having deleted the transactions already processed in the user input.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
4	4		Checkpoint frequency

Multi-entity User Input

Multi-purpose Line (Line VC, VG, VE, VO):

The VC-code line is used for calling Parameterized Input Aids and for assigning Comments to an Entity or some description lines.

Insertion of comments (VC line)

- a line which contains the type and code of the concerned entity and the line number,
- a line which contains the comments in column 4 and the character '*' in column 80 (value for continuation line).

Call of an Input Aid (VC and VZ lines)

- one VC line only is needed. It contains the type and code of the entity concerned, the line number if it is a Description and the value 'I' in the type of line column as well as the code of the Input Aid.
- a VZ line per variable part of the called Input Aid (see the following section, Parameterized Input Parameterized Input Aids/Variable Parts), the line subnumber and the description value.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'VC'	Line code for a 'GC' screen
		'VG'	Line code for a 'GG' screen
		'VE'	Line code for a 'GE' screen (call of a P.I.A. not possible in this screen)
		'VO'	Line code for a 'GO' screen
4	2		Entity type receiving the Comments
6	30		Entity code
36	3		Line number
39	3		Number of the commented line

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
42	1		Line type
43	6		Code of called P.I.A.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'VC'	Line code for a 'GC' screen
		'VG'	Line code for a 'GG' screen
		'VE'	Line code for a 'GE' screen (call of a P.I.A. not possible in this screen)
		'VO'	Line code for a 'GO' screen
4	60		Comment line
80	1		Continuation line
		'*'	This value must be entered to indicate a continuation line.

Parameterized Input Aids/Variable Parts (Line VZ):

The access line used for entering the contents of the variable parts is 'VZ'.

The structure of the VZ line must copy the P.I.A.'s Description one. The variable parts follow each other. There are no delimiters. The resolution includes the maximum length of each parameter defined.

Note

This line code always comes after a VC line (call of P.I.A.).

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'VZ'	
4	2		Number of parameter cards in a P.I.A
6	20		Printed label for level n
			This field contains the fixed part of a P.I.A. line as displayed when the P.I.A. is called.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			Its contents depend upon the TYPE OF P.I.A. LINE.
			The label is not justified (to be next to the value, it must be right-justified).
			On P.I.A. lines to be generated (value 'G' in the LINE GENERATION OPTION field on the P.I.A. Description (-D) screen), each instruction must be left-justified, and, if it does not fit on a single line, its continuation must begin with at least one 'blank' character.
26	40		DESCRIPTION / SECOND PART
			This field is specific to a P.I.A. call.
			With value 'C2' in the OPERATION CODE field, the cursor automatically tabs to the first position of this field.
			This field is initialized with blanks (default value) or with the value specified in the INITIAL VALUE field for a Standard PIA description line (Type = 'blank').
			If symbolic parameters have been defined on the P.I.A. Description (-D), they may be entered in this field. They will be replaced by their corresponding value, and will remain displayed on the right of the screen.

Call of Instances via Relations (Line QR):

The access line used for the call of instances via Relations is 'QR'.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'QR'	
4	2		Entity type receiving the Comments
6	2		Meta Entity Type
			It is an alphanumeric code entered upon creation and which characterizes the Meta Entity in all its uses (two Meta Entities cannot have the same type).
			The type cannot be modified if User Entities have already been defined for this ME.
			When used to define or describe a User Entity, this type is preceded by the '$' character (example: if the 'JOB' ME type is 'JO', the User Entities are referenced by '$JO.....').

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
8	30		Entity code (30 characters)
38	3		Line number
			Numeric. You are advised to begin with line number '100' and then number them in intervals of 20. This facilitates subsequent line insertions, as necessary.
			This field is alphanumeric if you generate a customized SQL access. In this case, you can enter letters in the 'LIN' field. You can then create more than the '1000' lines initially available.
41	6		User Relation code
47	30		Code of called entity (30 charac.)

Entity Update Lock (Line R):

The access line used to lock the update of entities is 'R'.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		'R'	
3	2		Entity type
			This field is used to specify the type of entity to which one or more keywords are assigned.
		'K1'	Model Entity.
		'S'	Text.
		'C'	Data Element.
		'A'	Data Structure.
		'2'	Segment.
		'V1'	Parameterized Input Aid.
		'L1'	Database Block.
		'H'	Screen.
		'B'	Report.
		'0'	Program.
		'U'	User Manual.
		'W1'	Volume.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		'Y1'	User Entity.
		'Y3'	Client Meta-Entity.
		'tt'	tt User Entity. Used for updating keywords of tt User Entities.
		'Y5'	User-Defined Relation.
5	2		Meta Entity Type
			It is an alphanumeric code entered upon creation and which characterizes the Meta Entity in all its uses (two Meta Entities cannot have the same type).
			The type cannot be modified if User Entities have already been defined for this ME.
			When used to define or describe a User Entity, this type is preceded by the '$' character (example: if the 'JOB' ME type is 'JO', the User Entities are referenced by '$JO.....').
7	30		Entity code
37	36		Entity name/comments
73	8		User code

Search by Keywords (Line G):

'G' is the access line used to define and assign explicit keywords.

On a first line, you find the type and code of the entity concerned.

Keywords (55 characters) are entered on a second line, a continuation line (identified by the '*' character at the end of the line).

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'G '	
4	2		Entity type receiving the Comments
6	30		Entity code
36	1		Call type
		'$'	Used to update keywords for User Entities.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'G '	
4	55		Keywords
80	1		Continuation line
		'*'	This value must be entered to indicate a continuation line.

Data Elements

Definition (Line C):

'C' is the access line used to define an Element.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		'C'	
3	6		Element code
9	1		Type (property, element or alias)
10	36		Data Element name
46	1		Type of format
		'I'	Internal format.
47	10		Data Element internal format
74	1		Element internal use
75	6		Code of parent Data Element

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		'C'	
3	6		Element code
9	1		Type (property, element or alias)
10	36		Data Element name

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
46	1		Type of format
		'E'	Input format.
47	10		Conversational format
74	1		Element internal use
75	6		Code of parent Data Element

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		'C'	
3	6		Element code
9	1		Type (property, element or alias)
10	36		Data Element name
46	1		Type of format
		'S'	Output format.
47	27		Output Format
74	1		Element internal use
75	6		Code of parent Data Element

Description (Line E):

'E' is the access line used to describe an Element.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		'E'	
3	6		Element code
9	3		Line number
12	1		Line type
13	1		Skip or action type
			Numeric

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
14	13		Data Element value
27	54		Data Element value - Meaning

Model Objects

Definition (Line K1):

The access line used to define a model entity, model relation or model F.I.C. is 'K1'.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'K1'	
4	6		Object code
10	36		Name of the object
46	1		Type of the object
		'O'	Object,
		'R'	Relationship,
		'C'	Functional Integrity Constraint (F.I.C.).
47	9		Number of instances
			Numeric
56	6		Code of the implied Relation
			This field is used for the definition of an F.I.C.
62	6		Parent object code
68	10		Object comment

Call of Properties in Object or Relat. (Line K3):

The line code used to call properties in an entity or a Model Relation is 'K3'.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		'K3'	
4	6		Object code
10	3		Line number
13	6		Element code
19	1		Identifier in Segment
20	3		Occurrences (COBOL "OCCURS" clause)
			Numeric
23	2		Number of Data Elements in a group

Model Relations

Definition (Line K1):

The access line used to define a model entity, model relation or model F.I.C. is 'K1'.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'K1'	
4	6		Object code
10	36		Name of the object
46	1		Type of the object
		'O'	Object,
		'R'	Relationship,
		'C'	Functional Integrity Constraint (F.I.C.).
47	9		Number of instances
			Numeric
56	6		Code of the implied Relation
			This field is used for the definition of an F.I.C.
62	6		Parent object code
68	10		Object comment

Call of Objects in Relation or F.I.C (Line K2):

The access line code used to call entities in a Relation or a F.I.C. is 'K2'.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'K2'	
4	6		Model Relation code
10	3		Line number
13	6		Object code
19	7		Occurrence ranking (minimal)
26	7		Occurrence ranking (maximal)
33	7		Average occurrence ranking

Call of Properties in Object or Relat. (Line K3):

The line code used to call properties in an entity or a Model Relation is 'K3'.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'K3'	
4	6		Object code
10	3		Line number
13	6		Element code
19	1		Identifier in Segment
20	3		Occurrences (COBOL "OCCURS" clause)
			Numeric
23	2		Number of Data Elements in a group

Model F.I.C.'s

Definition (Line K1):

The access line used to define a model entity, model relation or model F.I.C. is 'K1'.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'K1'	
4	6		Object code
10	36		Name of the object
46	1		Type of the object
		'O'	Object,
		'R'	Relationship,
		'C'	Functional Integrity Constraint (F.I.C.).
47	9		Number of instances
			Numeric
56	6		Code of the implied Relation
			This field is used for the definition of an F.I.C.
62	6		Parent object code
68	10		Object comment

Call of Objects in Relation or F.I.C (Line K2):

The access line code used to call entities in a Relation or a F.I.C. is 'K2'.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'K2'	
4	6		Model Relation code
10	3		Line number
13	6		Object code
19	7		Occurrence ranking (minimal)
26	7		Occurrence ranking (maximal)
33	7		Average occurrence ranking

Data Structures

Definition (Line A):

'A' is the access line used to define a Data Structure.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		'A'	
3	2		Data Structure code
5	30		Data Structure label
35	44		Data Structure comment
79	1		Type
80	1		File reporting option
		'O'	file descriptions will include vet and update markers. This option is to be used only for files with vets, update markers, fields with variable repetitions, or with initial values. It is mandatory for generating error messages.
		'N'	File descriptions will not include vet and update markers. In this case, field lengths and addresses in the record will be indicated (default option)
		'E'	file descriptions will be presented in their input format with addresses , lengths, and initial values of the fields in the record
		'I'	file descriptions will be presented in internal format with addresses, lengths, and initial values of the fields in the record

Segments

Definition (Line 2):

'2' is the access line used to define a Segment.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'2'	

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
4	4		Segment code
8	1		Type of Segment definition line
		'L'	
12	10		Structure Code value/Data Element
22	6		Code of action code element
			In the BATCH SYSTEMS DEVELOPMENT FUNCTION:
			Enter the DATA ELEMENT CODE for the element used to identify the transaction type. The System will generate validation logic appropriate for creation, modification, deletion and implicit action codes, as well as user-defined transaction types. Six values are associated with this code. Validation and updates are automatic for these six values:
			. transaction 1 creation, . transaction 2 modification, . transaction 3 deletion, . transaction 4 modification . transaction 5 modification, . transaction 6 modification.
			If there is no ACTION CODE ELEMENT, this field remains blank, and the transaction type is a modification. In this case, presence specifications for the segment are entered in the MOD-4 : ACTN CODE VALUE / SEG PRES. field, and for the elements, in the MOD-4 field on the Call of Elements (-CE) screen.
			The CODE OF ACTION CODE ELEMENT and the values must be entered on only one segment of the data structure, preferably on the common part '00'.
28	5		Create: Actn code value / Seg pres.
			(Specific to the Batch Systems Development function).
			ACTION CODE VALUE:
			On the '00' segment, enter the value that stands for "create" for this file: Example: 'ADD'. Note: for alphabetic characters use quotes.
			SEGMENT PRESENCE:
			On the non-00 segments, enter the presence specifications for the individual segment.
		'O'	Obligatory: the segment must be present on a "create"
		'I'	Invalid: the segment must not be present on a "create"
		'F'	Optional (default).
33	5		Modify: action code value/ Seg pres.
			(Specific to the Batch Systems Development function).

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			ACTION CODE VALUE:
			On the '00' segment, enter the value that stands for "modify" for this file: Example: 'CHG'. Note: for alphabetic characters use quotes.
			SEGMENT PRESENCE:
			On the non-00 segments, enter the presence specifications for the individual segment.
		'O'	Obligatory: the segment must be present on a "modify"
		'I'	Invalid: the segment must not be present on a "modify"
		'F'	Optional (default)
38	5		Delete: actn code value / Seg pres.
			(Specific to the Batch Systems Development function).
			ACTION CODE VALUE:
			On the '00' segment, enter the value that stands for "delete" for this file: Example: 'DEL'. Note: for alphabetic characters use quotes.
			SEGMENT PRESENCE:
			On the non-00 segments, enter the presence specifications for the individual segment.
		'O'	Obligatory: the segment must be present on a "delete"
		'I'	Invalid: the segment must not be present on a "delete"
		'F'	Optional (default).
43	5		Mod-4: actn code value / Seg pres.
			(Specific to the Batch Systems Development function).
			ACTION CODE VALUE:
			On the '00' segment, enter the value that stands for implicit action codes - (creates or modifications). Note: for alphabetic characters use quotes.
			SEGMENT PRESENCE:
			On the non-00 segments, enter the presence specifications for the individual segment.
		'O'	Obligatory: the segment must be present.
		'I'	Invalid: the segment must not be present.
		'F'	Optional (default).
48	5		Mod-5: actn code value / Seg pres.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			(Specific to the Batch Systems Development function).
			ACTION CODE VALUE:
			On the '00' segment, enter the value that stands for this user-defined action. Note: for alphabetic characters use quotes.
			SEGMENT PRESENCE:
			On the non-00 segments, enter the presence specifications for the individual segment.
		'O'	Obligatory: the segment must be present.
		'I'	Invalid: the segment must not be present.
		'F'	Optional (default).
53	5		Mod-6: actn code value / Seg pres.
			(Specific to the Batch Systems Development function).
			ACTION CODE VALUE:
			On the '00' segment, enter the value that stands for this user-defined action. Note: for alphabetic characters use quotes.
			SEGMENT PRESENCE:
			On the non-00 segments, enter the presence specifications for the individual segment.
		'O'	Obligatory: the segment must be present.
		'I'	Invalid: the segment must not be present.
		'F'	Optional (default)
58	1		Create: segment presence
59	1		Modify: segment presence
60	1		Delete: segment presence
61	1		Mod-4 : segment presence
62	1		Mod-5 : segment presence
63	1		Mod-6 : segment presence
64	4		Occurs in Table
68	9		Estimated number of instances
			Numeric
77	1		Continuation line indicator

Description (Line 3):

'3' is the access line used to call Elements into a Segment.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		'3'	
3	4		Segment code
7	3		Line number
10	6		Element code
16	18		Element short name
34	10		Data Element internal format
44	1		Element internal use
45	3		Occurrences (COBOL "OCCURS" clause)
			Numeric
48	2		Number of Data Elements in a group
50	1		Identifier in Segment
51	1		Creation
52	1		Modification
53	1		Deletion
54	1		Type 4
55	1		Type 5
56	1		Type 6
57	1		Class (alpha/numeric)
58	1		Operators (and/or)
59	1		Negation (NOT)
		'N'	Negation ('NOT' is generated).
		blank	No negation.
60	1		Type: validation, update, values
61	10		Values / sub-function code
71	2		Update target / first part
73	2		Update target / second part
75	6		Update target / last part

Pactables Sub-Schemas and Sub-Systems (Line 21):

The line code used to define all sub-schemas and sub-systems of a Table is '21'.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'21'	
4	4		Segment code
8	1		Type of Segment definition line
		'S'	Sub-schema definition.
		'Y'	Sub-system definition.
9	1		Sub-schema / sub-system number
10	30		Sub-schema/sub-system name
40	4		Occurs in Table

Reports

Definition (Line B):

'B' is the line code used to define a Report.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		'B'	
3	3		Report Code
6	30		Report name
36	36		Comments
72	1		Nature code
73	1		Type
74	3		Line length (maximum)
			Numeric
77	2		No. of digits left of the decimal
			Numeric

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
79	2		No. of digits right of the decimal
			Numeric

Report Layout Description (Line 4):

'4' is the line code used to describe a Report layout.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		'4'	
3	3		Report Code
6	2		Line number
8	2		Constant part number
			Numeric
10	1		Number of printed literals part
11	1		Line skip/page break
			Numeric
12	1		Char. set option: special printer
15	66		Edition label

Report Characteristics Description (Lines 5, E):

Batch Form '5' (type E) is used to describe the report characteristics.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		'5'	
3	3		Report Code
14	1		Line type
		'E'	Default value
15	3		Length of the variable part

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
24	1		Option
		'Blank'	Print options are generated according to the hardware variant indicated at the library level.
			In the case of conversion libraries, the print options are automatically reformulated according to the library variant.
		'N'	Prohibits any automatic reformulation of the print option, in a conversion library.
		'*'	Generation of 'WRITE BEFORE' statement.
25	2		Lines per page
			PURE NUMERIC FIELD
			Default option: 60.
27	4		No. of instances in category table
			PURE NUMERIC FIELD
			Enter the number of positions to allocate to store the different categories in the report (upon generation).
		'100'	Default.
		'0000'	Rather than using the category table to control the organization of printing the categories, the categories are printed directly.
			Note: If the number of positions is higher than 1000, the table is not generated.
31	2		Section priority
			This field is used with hardware requiring program segmentation due to small memory capacity. For information, consult a COBOL manual.
			Generates a segment type overlay between print functions in a program. It should only be used if input data structures to print programs are sorted by report code and if the COBOL variant is ANSI. Priorities less than 50 generate an overlay only in association with the 'SEGMENT LIMIT' clause, to be inserted in the ENVIRONMENT DIVISION.
33	13		Comments
			The comment entered on the screen top refers to the whole report. Comments entered on the screen body normally refer to the individual lines.
46	35		Condition of execution

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			The conditions indicated in the screen top are relevant to the execution of the Report.
			The conditions indicated in the screen body are relevant to the execution of the Category of the Report.
			Input format:
			Use the COBOL format to enter conditions but do not enter 'IF', nor GO TO, and do not enter any period.

List of Categories (Line 5):

'5' is the line code used to describe the report categories.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		'5'	
3	3		Report Code
6	2		Category
			(maximum of 39 lines per category.)
		'AB to ZY'	The value entered here is used to differentiate categories from one another. Report lines are grouped together according to the conditions under which they will be printed (totaled, etc...).
			Leaving gaps in the category sequence will facilitate future modifications.
			Categories containing a detail line with elements to be totaled - (TYPE OF LINE = '*' or 'T'):
			.can only contain one detail line,
			.cannot contain a total line,
			.cannot be repetitive,
			.can contain other ordinary lines.
			Categories used for the lines containing the totals - (TYPE OF LINE = '0' to '9'):
			.can contain several total lines,
			.cannot have a detail line,
			.cannot be repetitive,

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			.can contain other ordinary lines.
		'ZZ'	Prohibited.
		'AA'	Not recommended.
8	3		Line number
14	1		Type of line
			This field is used to identify the type of category.
			To designate a Header, repetitive area, or Footer:
		'A'	This value applies to repetitive categories only. This indicates the first line of a top-of-page category (header). Headers are automatically printed at the top of each page of a report. They are also printed when the repetitive category lines exceed the number of lines per page allowed for the report, causing a new page to be printed.
		'E'	In Batch forms, line which describes the Report (general characteristics and condition).
		'I'	Indicates the first line of a category printed several times (repetitive category). This value causes the generation of a subscript which controls the number of repetitions. This number may be fixed or variable.
			For a fixed number:
			.enter a number in the TOTALING LINE INDICATOR field
			For a variable number:
			.enter a three-character code in the TOTALING LINE INDICATOR field. (The code was defined on the Work Areas (-W) screen for use as the subscript field. Procedural code is used to move in the values.) OR .use the standard PACBASE index (Jddrcc), generated for the category: Note: ddr = REPORT CODE, cc = CATEGORY OF REPORT (repetitive) See SOURCE FIELD - LAST PART on the Report Call of Elements (-CE) screen, with value '*cc'.
		'Z'	This value applies to repetitive categories only. This indicates the first line of an end-of-page category (footer). Footers are automatically printed when the repetitive category lines exceed the number of lines per page allowed for that report.
			To identify detail lines with fields to accumulate:

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		'*'	This indicates a detail line containing fields whose values are to be accumulated for totaling. The lines will be printed in the report. Note: The data elements to total are identified on the Report Call of Elements screen by entering 'T' in OPERATION ON SOURCE FIELD. All elements are conditioned by report category. (See Subchapter "CALL OF DATA ELEMENTS (-CE)".)
			A category containing a detail line: . can contain only one detail line, . cannot contain a total line, . cannot be iterative, . can include other ordinary lines.
			The logic for data elements to be totaled is generated only if the conditions specified for the '*' line category are met.
		'T'	Same as '*', but the category containing this line is not to be printed.
			Note: For information concerning other lines that may or may not be included with lines of this type, see CATEGORY OF REPORT.
			One program may use several reports. There can only be 12 '*' and 'T' type lines (combined) per program.
			To identify lines displaying accumulated totals:
		'0'	Indicates a line for Grand Totals. Note: Grand Totals may only be requested if there is at least one Total at a control break level. At least one control break has to be specified for a file on the -CD screen.
		'1 to 9'	Indicates a line for totaling at the control break level corresponding to this value.
			A category containing a total line: . may contain several of them, . cannot contain a detail line, . cannot be iterative, . can include other ordinary lines.
			See CATEGORY OF REPORT for information on other lines that may or may not be included in a category with totaling-type lines.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			NOTE: A detail line may be defined in a different report. For example, a summary report based on accumulations from other reports may be needed. This can be done using the following technique: The STRUCTURE NUMBER assigned to the detail line of the other report is not used on the summary report's Call of Elements screen, and on its Description (-D) screen, the TYPE OF LINE value is entered and the TOTALING LINE INDICATOR will be comprised of the LAST CHARACTER OF REPORT CODE of the report containing the detail line, followed by its STRUCTURE NUMBER. Only the totaled data elements will be printed, at the designated control break level.
15	3		Characteristics
			On a line that has fields being totaled (TYPE OF LINE values '0' to '9'), which has a detail line described in a different report, enter the following:
			.first character: LAST CHARACTER OF REPORT CODE of the report containing the description,
			.2nd and 3rd characters: STRUCTURE NUMBER.
			On the first line of a repetitive category (TYPE OF LINE = 'I'), this value causes the generation of a subscript which controls the number of repetitions. This number may be fixed or variable.
			For a fixed number:
			.enter an absolute number value.
			For a variable number:
		'blank'	.enter the three character code defined on the Work Areas (-W) screen for use as the subscript field. (The values are determined via Procedural Code.) OR .use the standard PACBASE index (Jddrcc), generated for the category.
18	2		Structure number
			Numeric
20	2		Constant part number
			Numeric
22	2		Line skip/page break
			Numeric
24	1		Line skip type
27	2		Function code
29	2		Sub-function code

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
33	13		Comments
			The comment entered on the screen top refers to the whole report. Comments entered on the screen body normally refer to the individual lines.
46	35		Condition of execution
			The conditions indicated in the screen top are relevant to the execution of the Report.
			The conditions indicated in the screen body are relevant to the execution of the Category of the Report.
			Input format:
			Use the COBOL format to enter conditions but do not enter 'IF', nor GO TO, and do not enter any period.

Description of Structures (Line 6):

'6' is the line code used to call Elements into Structures.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		'6'	
3	3		Report Code
6	2		Structure number
			Numeric
8	3		Starting address (column number)
			Numeric
11	1		Data element line number
12	6		Element code
18	2		Continuation of D.S. Description
		'**'	Enter '**' to specify a continuation line, in which you indicate the continuation of a condition.
20	14		Output Format
			(Default option: INTERNAL FORMAT)

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			This is the format of a data element as it is used in a printed report, or in a screen as a display field. It is automatically transferred in the description of printed reports, screens and segments.
			It must be coded like a COBOL picture. USAGE is always DISPLAY.
			In previous versions, this field was used to generate the BLANK WHEN ZERO clause, which may be displayed in this field.
			When creating or updating a data element, the BLANK WHEN ZERO CLAUSE field must be used for this purpose.
			For data elements representing a date, it is possible to assign a symbolic format:
			Display type formats (input):
		'D'	Without century (picture (x6)).
		'C'	With century (picture (x8)).
			Internal type formats:
		'I'	Without century (picture x(6)).
		'S'	With century (picture x(8)).
			Extended type formats (output) (with slashes):
		'E'	With century (picture x(8)).
		'M'	With century (picture x(10)).
		'G'	Gregorian format (picture x(10)).
		'T'	TIME format.
		'TS'	TIMESTAMP format
			PACMODEL function: This field may be omitted for a property.
			For details on the use of the formats with the various types of Database Blocks, see the summary tables in chapter "COLUMNS: DATA ELEMENTS" of the "Relational SQL Database Description" Reference Manual.
34	1		Operation on source field
35	1		Working-Storage Prefix of Source
36	2		Source field - first part
38	2		Source field - second part

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
40	6		Code of source field
46	3		Source field - last part
49	32		Execution condition

On-Line Screens

Definition (Line H):

'H' is the line code used to define a Dialogue or a Screen information (name, number of lines and columns, etc.), and a second part, which contains:

- With a blank in the continuation field: the attributes, documentation call fields (PFkeys or characters), initialization character for entry fields;
- With '*' in the continuation field: the external name of the program, the external name of the map, the transaction code.

Usually, only one 'H' line code with the attributes is necessary to define a dialogue and only one 'H' line code with the external names is necessary to define a screen: in general, a screen takes on the attributes defined at the dialogue level.

However, both layout formats of line code 'H' can be entered to define a Dialogue or a Screen.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'H '	
4	2		Dialogue code
6	4		Screen code within the dialogue
10	30		Dialogue or screen name
40	2	NUMER.	Screen size - number of lines
			Numeric
42	3	NUMER.	Screen size - number of columns
			Numeric
45	1		Label type
46	2	NUMER.	Number of tabs per line

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			Numeric
48	2		Transactional language variant
50	1		Optional Command Lines Set/BEFORE
51	1		Control cards in front of map
52	1		Optional Command Lines Set/AFTER
53	1		Control cards in back of map
54	1		Intensity attribute - label
55	1		Intensity attribute - display field
56	1		Intensity attribute - input field
57	1		Intensity attribute - error message
58	1		Intensity attribute-erroneous field
59	1		Color attribute - label
60	1		Color attribute - display field
61	1		Color attribute - input field
62	1		Color attribute - error message
63	1		Color attribute - erroneous field
64	1		Presentation attribute - label
65	1		Presentation attribute-display field
66	1		Presentation attribute - input field
67	1		Presentation attribute-error message
68	1		Presentation att. - erroneous field
70	2		Help character: screen help
72	2		Help character: data element help
74	1		Initialization character: variables
78	2		Screen type
80	1		Continuation line

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'H '	
4	2		Dialogue code

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
6	4		Screen code within the dialogue
10	30		Dialogue or screen name
40	2	NUMER.	Screen size - number of lines
			Numeric
42	3	NUMER.	Screen size - number of columns
			Numeric
45	1		Label type
46	2	NUMER.	Number of tabs per line
			Numeric
48	2		Transactional language variant
50	1		Optional Command Lines Set/BEFORE
51	1		Control cards in front of map
52	1		Optional Command Lines Set/AFTER
53	1		Control cards in back of map
54	8		External name of program
62	8		External name of map
70	8		Transaction code
78	2		Screen type
80	1		Continuation line

Dialog Complement (Line H3):

'H3' is the line code used to enter the Dialogue Complement. It must be preceded by line code 'H', which specifies the Dialogue Code.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'H3'	
4	2		Common area - data structure code
6	1		Organization
7	8		External name of error message file
15	4		First screen code of the dialogue

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
19	6		Database Block code
25	4	NUMER.	Complementary common area length
			Numeric
29	47		Options

Description (Line I):

'I' is the line code used to describe a screen.

Note :

It must be preceded by a line code H which specifies the dialogue Code.

On the lines codes of screens description (I-type line code), enter the ? character in the column 31 to blank out the 'label type' field.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		'I'	
3	3		Line number
6	6		Element code
12	1		Positioning type
13	2	NUMER.	Line number positioning
			Numeric
15	3	NUMER.	Column number positioning
			Numeric
18	1		Nature of the data element
19	1		Label type
20	1		Intensity attribute - label
21	1		Intensity attribute - data
22	1		Presentation attribute - label
23	1		Presentation attribute - data
24	1		Color attribute - label
25	1		Color attribute - data

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
26	1		Cursor default position/skip option
27	2	NUMER.	Horizontal repetitions number
29	2	NUMER.	Vertical repetitions number
31	1		Presence validation of data element
32	1		Validation conditions/set variables
33	1		Update option
34	4		Update target: segment code
38	6		Update target / last part
44	1		Working-Storage Prefix of Source
45	4		Source segment code
49	6		Code of source field
60	2	NUMER.	Level

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		'I'	
3	3		Line number
6	6		Element code
12	1		Positioning type
13	2	NUMER.	Line number positioning
			Numeric
15	3	NUMER.	Column number positioning
			Numeric
18	1		Nature of the data element
19	1		Label type
20	1		Intensity attribute - label
21	1		Intensity attribute - data
22	1		Presentation attribute - label
23	1		Presentation attribute - data
24	1		Color attribute - label
25	1		Color attribute - data

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
26	1		Cursor default position/skip option
27	2	NUMER.	Horizontal repetitions number
29	2	NUMER.	Vertical repetitions number
31	1		Type of literal
			Defines the contents of the next field, which is displayed on the Call of Elements (-CE) with OPERATION CODE 2.
		blank	The field contains a fixed label value.
		'I'	The field contains an initial value automatically displayed when the Screen is invoked.
		'P'	The field contains a presentation value used for the Screen simulation only.
		'A'	This value indicates that the following label is made up of one character repeated more than 30 times.
			INPUT EXAMPLE:
			LABEL
			T LITERALS
			A 045-
			The corresponding label is a line of 45 dashes.
			IBM 36, IBM 38, IBM AS/ 400:
		'Y'	This value specifies that the next field contains an INDICATOR number for attribute positioning.
32	30		Displayed literal

Call of Segments (Line H2):

'H2' is the line code used to call segments into a screen.

It must be preceded by a line code 'H' which specifies the Screen Code.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'H2'	
4	1		Category indicator (screen)
5	4		Segment code in program

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			This group column contains the following elementary columns:
			CDSTPG (Code of Data Structure in Program), CRECPG (Code of Record in Program)
9	2		Line number
11	1		Access mode
		'S'	Sequential (default option).
		'R'	Random - Direct (indexed sequential organization only).
		'D'	Dynamic (VSAM files only - 'V' organization)
12	1		Use in reception
13	1		Use in display
14	4		Preceding segment code
18	14		Access key source
			This group column contains the following elementary columns:
			CSEGSR (Code of Source Segment), CDELSR (Code of Source Data Element).
32	6		Element code
38	1		Control break indicator for display
39	1		Organization
40	1		Generated description type
41	8		External name of the file
49	2		Data Structure code
51	2		Code
53	1		Sub-schema / sub-system number
54	2	NUMER.	Level

Call of Macro-Structures (Line M):

Macro-structures are called using the line code 'M'.

Since it contains no program or screen code, this line must always be preceded by a program or screen definition line.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		'M'	
3	2		Line number
5	1		Expansion option for Macro-Struct.
		blank	The macro-structure lines are expanded in the calling programs during the update
		'N'	No expansion of macro-structure lines during the update
6	1		Delimiter of parameter values
7	6		Macro-structure code
13	50		Parameter identifier
80	1		Continuation line

Program Beginning Insertions (Line D):

The 'Beginning of Program' is modified using the line code 'D'.

Since it contains no program or screen code, this line must always be preceded by a program or screen definition line.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		'D'	
4	2		Section to generate
7	2		Paragraph title
9	3		Line number
13	66		Instruction

Working Areas (Line 7):

The Work areas and Linkage areas are described using the line code '7'.

Since it contains no program or screen code, this line must always be preceded by a program or screen definition line.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		'7'	
5	2		Line beginning
7	3		Line number
10	1		Type of line or Data element format
			Type of line values:
		blank	Data entered in the LEVEL AND SECTION and WORK AREA DESCRIPTION fields are to be generated as entered.
		'-'	Continuation character for a literal.
		'*'	Comment. Data entered in the LEVEL AND SECTION and WORK AREA DESCRIPTION fields contain comments to be inserted into the generated Program (Table size is not considered as comment, see line's last field).
		'$'	This value appears in column 7 of the generated COBOL and the other Elements of the WORKING line appear as it is.
		'A'	Call of an eBusiness Application. This call is fully documented in the 'COBOL API User's Guide'.
		'F'	Call of a Data Structure.
			When 'F' is entered, the system responds with a formatted line which is used to facilitate data entry. The fields are the same as those used on the Call of Data Structures (-CD) screen for D.S. with ORGANIZATION = 'W' or 'L'.
			.Data Structure code in the Program.
			.Data Structure code in the Library.
			.Segment selection (enter the Segment code without an asterisk).
			(A segment code can only be renamed in batch).
			.Non-Printing Data Structure format (1 to 8).
			.Record type / Use within D.S. (I, E or S).
			.Level number (COBOL) of the record (1 to 5).
			.Organization.
			.Sub-schema number.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			Type 'F' '-W' lines are processed as Data Structure call lines (-CD) only for batch.
			If two Type 'F' '-W' lines referring to the same Data Structure (same Data Structure code in the Program) are separated, they will nevertheless be generated one after the other.
			Element format values:
		'E'	Use of the Input format of a Data Element.
		'I'	Use of the Internal format of a Data Element.
		'S'	Use of the Output format of a Data Element.
			For these format types, the presence of the Data Element in the Specifications Dictionary is checked. A cross-reference is established, which prohibits the deletion of the Data Element whenever the lines in which it is called have not been deleted themselves.
			If the Element does not exist in the Specifications Dictionary, the System sends a warning.
			When a global replacement is required (.C2), the Element is not checked but the cross-references will still be created.
			For these three format types, the entered data-name must therefore have the following format:
			W-DDSS-EEEEEE where:
			W = a working-storage prefix,
			DDSS = a given Data Structure and Segment code,
			EEEEEE = a Data Element code which exists in the Specifications Dictionary.
			The corresponding format is automatically attributed by the System.
			For IMS sub-monitors:
		'M'	Sub-monitor; enter the code of the sub-monitor in the LEVEL OR SECTION field.
		'C'	Call of a screen into the sub-monitor named above.
			Enter the SCREEN CODE of the screen belonging to the sub-monitor in the LEVEL OR SECTION field, followed by a space and a 'D' for Dynamic call or 'S' for Static.
			Example: C OOSCRN D
			Note: Enter one SCREEN CODE per 'C'-type line.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
11	17		Level or section
28	48		Work data declaration
76	5		Table size (occurs clause)

Procedural Code (Line P):

Procedural code is written using the line code 'P'.

Since it contains no program or screen code, this line must always be preceded by a program or screen definition line.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		'P'	
3	2		Function code
		'AA-99'	
		'$n'	In a macro-structure, the function code can be parameterized.
5	2		Sub-function
		'AA-99'	Sub-function code.
		'$n'	In a macro-structure, the sub-function code can be parameterized.
7	3		Line number
10	3		Operator
13	32		Operand
45	2	NUMER.	Level
47	2		Condition Type
49	32		Execution condition

Programs

Definition (Line 0):

'0' (zero) is the line code used to define a program.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		'0'	Zero
7	6		Program code
13	6		Code for sequence of generation
19	30		Program name
49	1		Type of COBOL
50	1		Order of insertion in COBOL Library
51	1		COBOL numbering and alignment option
54	1		SQL indicators generation with '-'
55	1		Optional Command Lines Set/BEFORE
56	1		Optional Command Lines Set/AFTER
57	8		COBOL program id
65	1		Programming mode
66	1		Type and structure of program
67	1		Type of presence validation
68	1		Program classification code

Call of Data Structures (Line 1):

'1' is the line code used for the 'Call of Data Structures'.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		'1'	
3	2		Data structure code in the program
5	2		Data Structure code
7	6		External name
13	1		Organization
14	1		Access mode
15	1		Recording mode
16	1		Opening mode

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
17	1		Unit type
18	5		Block size
			Numeric
23	1		Block size unit type
24	10		File status
34	6		Indexed Data Structure Access Key
			Required for indexed Data Structures: Enter the DATA ELEMENT CODE of the access key Element.
40	1	NUMER.	Number of control breaks
			Numeric
41	1		File matching level number
			Numeric
42	1		Usage
43	6		Element code
49	2		Resulting file data structure code
51	2		Source or error data structure code
53	1		Transaction control break level
59	4		Physical Unit Type
63	1		Unit Complement
64	9		Sort key / seg select / report codes
73	1		Format type
74	1		Selected description
75	1		Generated description type
76	1		Level
77	2		Line beginning
79	2		Continuation of D.S. description

Call of Macro-Structures (Line M):

Macro-structures are called using the line code 'M'.

Since it contains no program or screen code, this line must always be preceded by a program or screen definition line.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		'M'	
3	2		Line number
5	1		Expansion option for Macro-Struct.
		blank	The macro-structure lines are expanded in the calling programs during the update
		'N'	No expansion of macro-structure lines during the update
6	1		Delimiter of parameter values
7	6		Macro-structure code
13	50		Parameter identifier
80	1		Continuation line

Program Beginning Insertions (Line D):

The 'Beginning of Program' is modified using the line code 'D'.

Since it contains no program or screen code, this line must always be preceded by a program or screen definition line.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		'D'	
4	2		Section to generate
7	2		Paragraph title
9	3		Line number
13	66		Instruction

Working Areas (Line 7):

The working and linkage areas are described using the line code '7'.

Since it contains no program or screen code, this line must always be preceded by a program or screen definition line.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		'7'	
5	2		Line beginning
7	3		Line number
10	1		Type of line or Data element format
			Type of line values:
		blank	Data entered in the LEVEL AND SECTION and WORK AREA DESCRIPTION fields are to be generated as entered.
		'-'	Continuation character for a literal.
		'*'	Comment. Data entered in the LEVEL AND SECTION and WORK AREA DESCRIPTION fields contain comments to be inserted into the generated Program (Table size is not considered as comment, see line's last field).
		'$'	This value appears in column 7 of the generated COBOL and the other Elements of the WORKING line appear as it is.
		'A'	Call of an eBusiness Application. This call is fully documented in the 'COBOL API User's Guide'.
		'F'	Call of a Data Structure.
			When 'F' is entered, the system responds with a formatted line which is used to facilitate data entry. The fields are the same as those used on the Call of Data Structures (-CD) screen for D.S. with ORGANIZATION = 'W' or 'L'.
			.Data Structure code in the Program.
			.Data Structure code in the Library.
			.Segment selection (enter the Segment code without an asterisk).
			(A segment code can only be renamed in batch).
			.Non-Printing Data Structure format (1 to 8).
			.Record type / Use within D.S. (I, E or S).
			.Level number (COBOL) of the record (1 to 5).
			.Organization.
			.Sub-schema number.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			Type 'F' '-W' lines are processed as Data Structure call lines (-CD) only for batch.
			If two Type 'F' '-W' lines referring to the same Data Structure (same Data Structure code in the Program) are separated, they will nevertheless be generated one after the other.
			Element format values:
		'E'	Use of the Input format of a Data Element.
		'I'	Use of the Internal format of a Data Element.
		'S'	Use of the Output format of a Data Element.
			For these format types, the presence of the Data Element in the Specifications Dictionary is checked. A cross-reference is established, which prohibits the deletion of the Data Element whenever the lines in which it is called have not been deleted themselves.
			If the Element does not exist in the Specifications Dictionary, the System sends a warning.
			When a global replacement is required (.C2), the Element is not checked but the cross-references will still be created.
			For these three format types, the entered data-name must therefore have the following format:
			W-DDSS-EEEEEE where:
			W = a working-storage prefix,
			DDSS = a given Data Structure and Segment code,
			EEEEEE = a Data Element code which exists in the Specifications Dictionary.
			The corresponding format is automatically attributed by the System.
			For IMS sub-monitors:
		'M'	Sub-monitor; enter the code of the sub-monitor in the LEVEL OR SECTION field.
		'C'	Call of a screen into the sub-monitor named above.
			Enter the SCREEN CODE of the screen belonging to the sub-monitor in the LEVEL OR SECTION field, followed by a space and a 'D' for Dynamic call or 'S' for Static.
			Example: C OOSCRN D
			Note: Enter one SCREEN CODE per 'C'-type line.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
11	17		Level or section
28	48		Work data declaration
76	5		Table size (occurs clause)

Procedural Code (Line P):

Procedural code is written using the line code 'P'.

Since it contains no program or screen code, this line be must always be preceded by a program or screen definition line.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		'P'	
3	2		Function code
		'AA-99'	
		'$n'	In a macro-structure, the function code can be parameterized.
5	2		Sub-function
		'AA-99'	Sub-function code.
		'$n'	In a macro-structure, the sub-function code can be parameterized.
7	3		Line number
10	3		Operator
13	32		Operand
45	2	NUMER.	Level
47	2		Condition Type
49	32		Execution condition

COBOL Source Lines (Line FC):

Source Code is written using the line code 'FC'.

Since it contains no program code, this line must always be preceded by a program definition line.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'FC'	
4	2		Function code
		'AA-99'	
		'$n'	In a macro-structure, the function code can be parameterized.
6	2		Sub-function
		'AA-99'	Sub-function code.
		'$n'	In a macro-structure, the sub-function code can be parameterized.
8	3		Line number
11	3		Operator
14	67		Source line

Pure COBOL Source Lines (Line 9):

Pure COBOL Source Code (-9) lines may be entered on line code '9'.

Since it contains no program code, this line must always be preceded by a program definition line.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		'9'	
3	6		COBOL line number
9	1		Continuation line
10	65		COBOL instruction
75	6		End of COBOL line

Database Blocks (Hierarchical)

Definition (Line L1):

'L1' is the line code used to define a Database Block.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'L1'	
4	6		Database Block code
10	36		Name of the block
46	8		Database Block external name
54	2		Block type
56	8		External name of the schema
64	1		Control cards in front of block
65	1		Control cards in back of block
66	4		Version number

Description (Line L2):

'L2' is the line code used to describe a Hierarchical Database Block.

The same line code is used for the Descriptions of SOCRATE/CLIO sub-structures but only the following lines are filled in: the block code, the action code, the line number and, in the column reserved for the Model Relationship code, the code of the structure to which the sub-structure belongs.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'L2'	
4	6		Database Block code
10	3		Line number
13	4		Child segment code
17	4		Parent segment code

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
21	6		Model Relation code
27	1		Identifier in Segment
28	5		Estimated number: child/parent links
			Numeric
33	36		Comment/relation/key length
69	6		Path item (turboimage)
75	6		Sort path item (turboimage)

Database Blocks (Codasyl)

Definition (Line L1):

'L1' is the line code used to define a Database Block.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'L1'	
4	6		Database Block code
10	36		Name of the block
46	8		Database Block external name
54	2		Block type
56	8		External name of the schema
64	1		Control cards in front of block
65	1		Control cards in back of block
66	4		Version number

Description (Line L3):

'L3' is the line code used to describe CODASYL, DB2, and TANDEM Database Blocks.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
2	2		Line code
		'L3'	
4	6		Database Block code
10	3		Line number
13	1		TYPE
		'S'	Set.
		'*'	Continuation of a set.
			For a set with multiple members, the first MEMBER Segment is indicated on an 'S'-type line, the others on '*'-type lines.
		'R'	Record.
		'A'	Area.
14	6		Area or set code
20	4		Parent segment code
24	4		Child segment code
28	6		Model Relation code
34	5		Estimated number: child/parent links
			Numeric
39	36		Comment/relation/key length

Database Blocks (Relational-SQL)

Definition (Line L1):

'L1' is the line code used to define a Database Block.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'L1'	
4	6		Database Block code
10	36		Name of the block
46	8		Database Block external name
54	2		Block type

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
56	8		External name of the schema
64	1		Control cards in front of block
65	1		Control cards in back of block
66	4		Version number

Description (Line L4):

'L4' is the line code used to describe a Relational/SQL Database Block.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'L4'	
4	6		Database Block code
10	3		Line number
13	1		Structure code SQL batch transact
14	1		SQL record type
		'P'	Tablespace (except for Interel RDBC, Interel RFM, Nonstop SQL, Sybase and SQL Server)
		'T'	Table
		'V'	View
		'I'	Index
		'A'	Alter Table: Column updating
		'K'	RDMS 1100: Primary Key (Processed with the generation of the table that precedes it.)
			DB2, Datacom/DB, SQL/DS, Oracle V6 and V7, DB2/2, DB2/6000, Sybase and SQL Server: Primary key (Processed with the generation through an ALTER TABLE command.)
		'J'	DB2, Datacom/DB, SQL/DS, Oracle V6 and V7, Sybase and SQL Server: Foreign key (Processed with the generation through an ALTER TABLE command.)
		'C'	Package (Oracle V7 only)
		'E'	Function (Oracle V7 only)
		'Q'	Procedure (Oracle V7, Sybase, SQL Server)

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		'R'	Oracle V7, Sybase and SQL Server: Trigger
15	18		Method external name
33	4		Segment code
37	1		Order
41	1		Key type
43	1		Type of generated transaction
44	6		Code of key data element no.1
50	1		Sort order 1
51	6		Code of key data element no.2
57	1		Sort order 2
58	6		Code of key data element no.3
64	1		Sort order 3
65	6		Code of key data element no.4
71	1		Sort order 4
72	6		Code of key data element no.5
78	1		Sort order 5

Database Blocks (Turboimage)

Definition (Line L1):

'L1' is the line code used to define a Database Block.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'L1'	
4	6		Database Block code
10	36		Name of the block
46	8		Database Block external name
54	2		Block type
56	8		External name of the schema
64	1		Control cards in front of block

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
65	1		Control cards in back of block
66	4		Version number

Description (Line L2):

'L2' is the line code used to describe a Hierarchical Database Block.

The same line code is used for the Descriptions of SOCRATE/CLIO sub-structures but only the following lines are filled in: the block code, the action code, the line number and, in the column reserved for the Model Relationship code, the code of the structure to which the sub-structure belongs.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'L2'	
4	6		Database Block code
10	3		Line number
13	4		Child segment code
17	4		Parent segment code
21	6		Model Relation code
27	1		Identifier in Segment
28	5		Estimated number: child/parent links
			Numeric
33	36		Comment/relation/key length
69	6		Path item (turboimage)
75	6		Sort path item (turboimage)

Texts

Definition (Line S):

'S' is the line code used to define a Text.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		'S'	
3	6		Text code
9	36		Text name
45	2		Type of text
47	2		Paragraph type

Description (Line T):

'T' is the line code used to describe a text.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		'T'	
3	6		Text code
9	2		Text paragraph
11	3		Line number
14	1		Type of Text line
			Section title:
			A section must always contain a title. The choice TttttttLT is used to consult the list of titles.
		'L'	Section title. It will NOT appear in a Volume.
		'K'	Section title. This line will appear in a Volume.
		'-'	Section title. This line will be underlined with the '-' (dash) character when a Volume is printed.
		'_'	Section title. This line will be underlined with the '_' (underscore) character when a Volume is printed.
		'='	Section title. This line will be underlined with the '=' character when a Volume is printed.
		'+'	Section title. This line will be underlined with the '+' character when a Volume is printed.
			Text Description line:

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		blank	Description line printed without additional skip (default option).
		('1' '9')	Number of lines to skip before the line text printing: the value '1' means new line, the value '2' inserts one line before the given line is printed, the value '3' inserts two lines, etc.
		'*'	PAGE skip before the given line is printed.
			Text Assignment: Text can be assigned to instances of other entities called in the TEXT DESCRIPTION LINE field. The assignment starts at the beginning of the section which contains the I-type line and terminates at the end of the text or after a J-type line. The assignment for one instances, all instances of a given entity or of all entities can be terminated. The '-AT" choice is used to visualize the texts assigned to the instance of an entity. Texts can be assigned to the following entities: 'B' (Database block), 'D' (Data structure), 'E' (Data element), 'F' (Meta-entity), 'I' (Input Aid), 'M' (Model entity), 'O' (Screen), 'P' (Program), 'Q' (User relation), 'R' (Report), 'S' (Segment), 'T' (Text), 'V' (Volume), '$' (User entity).
		'I'	Beginning of assignment. It starts at the beginning of the section which contains this line.
		'J'	Explicit end of assignment.
		'B'	Same as type 'I' plus possibility to enter codes of User Entity instances longer than 6 characters.
		'E'	Same as type 'J' plus possibility to enter codes of User Entity instances longer than 6 characters.
		'Y'	This code is used to create a link between this section of text and another text or section, i.e. 'refer to..'. The System displays the title of this text or section.
			For the referenced text:
			Choice -XT gives the list of texts referring to the whole text, Choice -LT gives the list of sections, each followed by the sections referring to it.
			Note: The L, I, J, B, E and Y Type lines are not printed in Volumes.
15	60		Text contents
75	6		Element code

Documents

Definition (Line W1):

'W1' is the line code used to define a Document.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'W1'	
4	6		Volume code
10	36		Volume name
46	1		Volume type
47	1		Title page option
48	1		Table of contents source
49	1		Table of contents placement
50	6		Text code
56	3		Report Code
59	3		Report code for font types
62	3		Report code for specific layout
65	1		Volume description organization mode

Description (Line W2):

'W2' is the line code used to describe a Document.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'W2'	
4	6		Volume code
10	2		Level 1 code
12	2		Level 2 code
14	3		Line number
17	1		Type of volume description line
18	1		Section level number

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
19	1		Line skip/page break
			Numeric
20	1		Character for title underlining
21	1		Print window
22	1		Alignment option
23	50		Title, printing opt. or entity sel.
73	4		Reference cursor

Parameterized Input Aids

Definition (Line V1):

'V1' is the access line used to define a P.I.A.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'V1'	
4	6		Input Aid code
10	36		Parameterized Input Aid name
46	1		Parameterized Input Aid type
		'C'	Comment
		'G'	Generation
		'O'	Option

Description (Line V2):

'V2' is the line code used to describe a P.I.A.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'V2'	
4	6		Input Aid code

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
10	3		Line number
13	1		Line type
14	20		Label
34	29		Initial value of P.I.A. line
63	3		Length
66	6		Reference Name
72	1		Line Option

Meta-Entities

Definition (Line Y1):

'Y1' is the line code used to define a Meta-Entity.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'Y1'	
4	6		Client Meta-Entity code
10	36		Client Meta-Entity label
46	2		Meta-Entity calling code

Detail Line Definition (Line Y6):

'Y6' is the line code used to define the UE detail lines of the Meta-Entity.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction Code
2	2		Line code
		'Y6'	
4	6		Client Meta-Entity code
10	2		Description type
12	1		Description type

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
13	30		Meta Entity Description Label
43	8		Subprogram code
51	1		Data storage mode
54	2		Parent description type

Description (Line Y2):

'Y2' is the line code used to describe a Meta-Entity.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'Y2'	
4	6		Client Meta-Entity code
10	2		Description type
12	3		Line number
15	6		Element code
21	2		Range
23	1		Element top nature
24	1		Uppercase top change
25	1		Element format top control
26	1		Presence top control
27	1		Value top control
28	6		User Relation Code
73	1		Parent identifier code
74	1		Identifier code called by Relation

User-Defined Relations

Definition (Line Y5):

'Y5' is the line code used to define a User-Defined Relation.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'Y5'	
4	6		Client User Relation
10	36		Client User Relation label
46	14		Client User Relation short label
60	1		Client User Relation type
61	3		Entity Type (3 characters)
			The authorized values are the Entity type values given in chapter "DAF Entities: Coding rules", subchapter "Tables" of the "DSMS Access Facility Tables" manual.
64	1		Deletion flag

Client User Entities

Definition (Line Y3):

'Y3' is the line code used to define a Client User Entity.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'Y3'	
4	2		Meta-Entity calling code
6	6		User Entity short code
12	2		Range
14	1		Transaction number for User Entity
		blank	First line
		'*'	Continuation line
15	66		User Entity Definition Transaction

Description (Line Y4):

'Y4' is the line code used to describe the detail lines of a Client User Entity.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'Y4'	
4	2		Description code
6	6		User Entity descr. short identifier
12	2		Range
14	1		Transaction number for User Entity
		blank	First line
		'*'	Continuation line
15	66		User Entity Definition Transaction

Extension User Entities

Definition (Line YC):

'YC' is the line code used to define an Extension User Entity.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'YC'	
4	2		Meta-Entity calling code
6	6		User Entity short code
12	2		Range
14	1		Transaction number for User Entity
		blank	First line
		'*'	Continuation line
15	66		User Entity Definition Transaction

Description (Line YD):

'YD' is the line code used to describe the detail lines of an Extension User Entity.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'YD'	
4	2		Description code
6	6		User Entity descr. short identifier
12	2		Range
14	1		Transaction number for User Entity
		blank	First line
		'*'	Continuation line
15	66		User Entity Definition Transaction

Thesaurus

Enrichment of the Thesaurus (Line G1):

'G1' is the access line used to document keywords (enrichment of the Thesaurus).

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		'G1'	
4	13		KEYWORD
17	1		Continuation line
18	1		Keyword description type
		'D'	Comments
		'S'	Synonym(s)
19	55		Keyword description

Library

Definition (Line X):

'X' is the line code used to define a Library.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	36		Library name
37	1		Date format indicator
38	1		Vertical character of frame
39	1		Stream OCLS/BEFORE
40	1		Stream OCLS/AFTER
41	2	NUMER.	Lines per page in documentation
			Numeric
43	1		Page skip
44	1		Comments Insertion Option
45	1		Modification of extracted lines
46	1		Optional Command Lines Set/BEFORE
47	1		Optional Command Lines Set/AFTER
48	1		Generation Language
49	1		Type of COBOL
50	1		Programming mode
51	1		Protection of extracted entities
57	1		Date Format in Generated programs
58	1		Decimal Point Presentation character
59	1		TP Monitor and Map Type
60	1		Generated COBOL formatting
61	1		Alphanumeric Delimiter
62	1		Horizontal character of frame
63	1		Century System Date
64	2	NUMER.	Reference Year for Century

UPDT - Description of Steps

Database consistency check: PTUBAS

Code		Type	Label
PAC7AR	Base dir.: AR	Input	Development Database data
PAC7AE	System - Skel dir.: AE	Input	Error messages
PAC7DS		Report	Validity report

.Return code:

This utility sends a return code 4 and causes an ABEND if the Database is inconsistent.

Transactions formatting: PACA05

Code	Physical name	Type	Label
PAC7AR	Database dir.: AR	Input	Development Database Data file
PAC7AN	Database dir.: AN	Input	Development Database Index file
PAC7AY	Database dir.: AY	Input	Development Database extension data
PAC7AE	System - Skel. dir.: AE	Input	Error messages
PACGGN	Admin. Database - Base dir.: AN	Input	Administration Database Index file
PACGGR	Admin. Database - Base dir.: AR	Input	Administration Database Data file
PACGGU	Admin. Database - Base dir.: GU	Input	Administration Database Users
PAC7MB	User input	Input	Update transactions
PAC7ME	Tmp dir.: WME	Output	Work file (length=372)
PAC7MV	Tmp dir. : WMV	Output	Formatted transactions (length=170, must be able to contain all input transactions plus the elementary delete transactions generated by the multiple delete transactions)
PAC7MW	Tmp dir. : WMW	Output	Work file

Update of the Development Database: PACA15

Code	Physical name	Type	Label
PAC7AR	Database dir.: AR	Output	Development Database Data file
PAC7AN	Database dir.: AN	Output	Development Database index
PAC7AY	Database dir.: AY	Output	Development Database extension
PAC7AJ	Journal dir.: AJ	Output	Development Database journal
PAC7AE	System - Skel. dir.: AE	Input	Error messages

Code	Physical name	Type	Label
PACGGN	Admin. Database - Base dir.: AN	Input	Administration Database Index file
PACGGR	Admin. Database - Base dir.: AR	Input	Administration Database Data file
PACGGY	Admin. Database - Base dir.: AY	Input	Administration Database Extension
PACGGU	Admin. Database - Base dir.: GU	Input	Administration Database users
PAC7DC	Base dir.: DC	Input	DSMS file of Development Database Elements
PAC7ME	Tmp dir.: WME	Input	Work file
PAC7MV	Tmp dir.: WMV	Input	Update transactions
PAC7RB	User dir. : RBA15	Output	UPDT erroneous transactions (length=80)
PAC7RY	User dir. : RYA15	Output	UPDP erroneous transactions (length=310)
PAC7IE	User dir. : IEA15	Report	Update report (length=132)
PAC7IF	User dir. : IFA15	Report	List of erroneous transactions (length=132)

The list of user transactions is preceded by a banner with the user code.

Return codes:

- 0: OK, no error
- 2: Warning
- 4: Critical error

UPDT - Execution Script

```
' -------------------------------------------------------
'      VISUALAGE PACBASE
'
' -------------------------------------------------------
'                 - BATCH UPDATE -
'
' -------------------------------------------------------
'
' REFER TO THE BATCH FORMS AND TO THE DESCRIPTION OF THE
' INPUT CORRESPONDING TO EACH ENTITY.
'
' INPUT :
'  -  USER IDENTIFICATION LINE (REQUIRED)
'     COL  2 : "*"
'     COL  3 : USERIDXX
```

```
'     COL 11 : PASSWORD
'     COL 28 : LANGUAGE CODE, USEFUL WHEN TRANSACTION ARE
'              NOT IN THE SAME LANGUAGE AS THE DATABASE.
'     COL 67 : "N"  DEFAULT VALUE WITH EXTRACTORS
'  -  COMMAND LINE
'     THE LIST OF ALL AVAILABLE VALUES FOR THE ENTITY
'     TO BE UPDATED IS FOUND IN REFERENCE MANUAL.
'
' ------------------------------------------------------
'
<job id=UPDT>

<script language="VBScript">
Dim MyProc
MyProc = "UPDT"
</script>

<script language="VBScript" src="INIT.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If

Call Msg_Log (Array("1022" , "PTUBAS"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
Call BvpEnv("PTUBAS","PAC7DS",Rep_USR & "\UPDTDSBAS.txt")
Call RunCmdLog ("BVPTUBAS")
WshVolEnv("RC") = Return
If Return = 4 Then
Call Msg_Log (Array("1051"))
End If
Call Err_Cod(Return , 0 , "PTUBAS")

Call Msg_Log (Array("1022" , "PACA05"))
'-----------------------------------
WshEnv("PAC7MB") = Fic_Input

'Example of Input File extracted from PACX :
' Call BvpEnv("PACA05","PAC7xx",RepT_USR & "\PACXxx.txt")
'With  RepT_USR is Global User Directory.

WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PAC7AY") = Rep_BASE & "\AY"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
Call BvpEnv("PACA05","PAC7MV",Rep_TMP & "\WMV.tmp")
Call BvpEnv("PACA05","PAC7ME",Rep_TMP & "\WME.tmp")
Call BvpEnv("PACA05","PAC7MW",Rep_TMP & "\WMW.tmp")
Call RunCmdLog ("BVPACA05")
```

```
Call Err_Cod(Return , 0 , "PACA05")

Call Msg_Log (Array("1022" , "PACA15"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AJ") = Rep_JOURNAL & "\AJ"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AY") = Rep_BASE & "\AY"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PACGGY") = Rep_ABASE & "\AY"
Call BvpEnv("PACA15","PAC7DC",Rep_BASE & "\DC")
WshEnv("SEMLOCK") = Rep_BASE & "\LO"
WshEnv("SEMADMIN") = Rep_ABASE & "\LO"
Call BvpEnv("PACA15","PAC7IE",Rep_USR & "\UPDTIEA15.txt")
Call BvpEnv("PACA15","PAC7IF",Rep_USR & "\UPDTIFA15.txt")
Call BvpEnv("PACA15","PAC7ME",Rep_TMP & "\WME.tmp")
Call BvpEnv("PACA15","PAC7MV",Rep_TMP & "\WMV.tmp")
Call BvpEnv("PACA15","PAC7RB",Rep_USR & "\UPDTRBA15.txt")
Call BvpEnv("PACA15","PAC7RY",Rep_USR & "\UPDTRYA15.txt")
Call RunCmdLog ("BVPACA15")
If Return = 2 Then
Call Msg_Log (Array("1061"))
End If
If Return = 4 Then
Call Msg_Log (Array("1060"))
End If
Call Err_Cod(Return , 4 , "PACA15")

Call Msg_Log (Array("1024"))
'-----------------------------------
Call DeleteFldr (Rep_TMP)

Call Msg_Log (Array("1023"))
'-----------------------------------
Wscript.Quit (Return)


</script>
</job>
```

Chapter 6. Pactables

GETD-GETA - Description Generators

GETD-GETA - Introduction

The Table Description Generator is the interface between the Specifications Dictionary and Pactables. For further information, refer to chapter 'General Introduction", subchapter "Introduction to the Pactables Facility" in the "Pactables" manual.

Its use is subject to a purchase agreement.

This interface extracts, from the VisualAge Pacbase Database, the descriptions of the Tables necessary for the operation of the Pactables Facility.

This extraction is executed via either the GETA or GETD procedure according to the installation environment of the Pactables Facility:

- GETA if the Dictionary and Pactables are running under the same environment.
- GETD if the Dictionary and Pactables are running under different environments. In this case, GETD processes a table description file which is the image of the file containing the table descriptions used by the Pactables Facility. As a result, this file must be initialized before the first GETD run, by:
  - either duplicating the description file of the Pactables Facility, if it exists,
  - or executing the initialization procedure (GETI) described in this chapter.

GETA or GETD provides an interface file which is used as input to the GETT procedure of the Pactables Facility. For further details, refer to the 'Pactables' manual.

Execution conditions

None with regard to the Specifications Database, which is only read by this procedure.

Abnormal execution

If the generation abends before the update of the table description file, the procedure can be restarted as it is once the error has been corrected.

If the generation abends during the update of the table description file, this file must be restored before the procedure is restarted.

GETD-GETA - User Input / Result

User input

A '*'-type line indicating the Library which contains the table descriptions.

Position	Length	Value	Meaning
2	1	'*'	Line code
3	8	uuuuuuuu	User code
11	8	pppppppp	User password
19	3	bbb	Library code
22	4	nnnn	Session number
26	1	t	Session status

One 'Z' line per generation or print request.

Position	Length	Value	Meaning
2	1	'Z'	Line code
5	4		Request code:
		'TGS '	Request for table descrip. generation
		'TDS '	Request for printing of table descr.
		'TLS '	Request for list of table descriptions
		'TAS '	Request for table deletion
		'TMS '	Request for modification of frozen table characteristics
		'TGC '	Request for comments generation
9	6	ssss	Segment code of table description to be extracted ('TGS ','TGC ')
		tttttt	Table code (other requests)
15	2	' '	Not significant
17	8	MMDDCCYY	Date from which the table description can be modified. (Optional)
25	8	MMDDCCYY	Date of description historical account for a G-type table. Default: last historical account.
		********	Table generation without hist. account
33	1		Data Element format type:

Position	Length	Value	Meaning
		' '	Internal format
		'E'	Input format
75	6	ttttt	Table number (if generating for a table other than that of the Segment's Definition in the Database).

For further information on this user input, please refer to the Pactables Reference Manual.

Note: Table keys cannot be modified. The generation requests for tables already defined and which involve such modifications are rejected.

Result

The output of the GETA procedure is a sequential file containing table descriptions, which will be used as input to the GETT procedure of the Pactables Facility.

GETD-GETA - Description of Steps

Extraction & update preparation: PACT40

Code	Physical name	Type	Label
PAC7AR	Database dir. : AR	Input	Development Database Data file
PAC7AN	Database dir. : AN	Input	Development Database Index file
PAC7AY	Database dir. : AY	Input	Development Database extension data
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database index
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database data
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database users
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7TD	Database dir. : TD	Input	Tables descriptions
PAC7MB	User input	Input	Requests for descriptions
PAC7MD	User dir. : MVGETD or MVGETA	Output	Update transactions for the Descriptions whose version is equal to or greater than 2.0
PAC7ET	User dir. : GETDETT40 or GETAETT40	Report	Output report

Code	Physical name	Type	Label
PAC7DD	User dir. : GETDDDT40 or GETADDT40	Report	Batch procedure authorization option

Return Codes :

- 8 : no batch procedure authorization.

Formatting of descriptions < V 2.0: PACT45

Code	Physical name	Type	Label
PAC7MD	User dir. : MVGETD or MVGETA	Input	Update transactions for the Descriptions whose version is equal to or greater than 2.0
PAC7ND	User dir. : NDGETD or NDGETA	Output	Update transactions for the Descriptions whose version is lower than 2.0

Update of table descriptions file: PACT50

(GETD procedure only)

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7AR	Database dir. : AR	Input	Development Database Data file
PAC7TD	Database dir. : TD	Input	Tables descriptions
PAC7MD	User dir. : MVGETD	Input	Update transactions
PAC7ET	User dir. : GETDETT50	Report	Update report

GETD - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'             - TABLE DESCRIPTIONS GENERATION -
'
' ------------------------------------------------------
'
' TABLE DESCRIPTION GENERATOR IS THE INTERFACE BETWEEN
' THE SPECIFICATIONS DICTIONARY AND VA PACTABLES.
' FOR FURTHER INFORMATION, REFER TO CHAPTER GENERAL
' INTRODUCTION SUBCHAPTER INTRODUCTION TO THE
' VA PACTABLES FACILITY IN THE VA PACTABLES MANUAL.
```

```
'
' GETD IF THE DICTIONARY AND VA PACTABLES ARE RUNNING
' UNDER DIFFERENT ENVIRONMENTS.
'
' -----------------------------------------------------
'
<job id=GETD>

<script language="VBScript">
Dim MyProc
MyProc = "GETD"
</script>

<script language="VBScript" src="INIT.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If

If  FSO.FileExists(Rep_BASE & "\TD") Then

Call Msg_Log (Array("1022" , "PACT40"))
'--------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PAC7AY") = Rep_BASE & "\AY"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7TD") = Rep_BASE & "\TD"
WshEnv("PAC7MB") = Fic_Input
Call BvpEnv("PACT40","PAC7MD",Rep_USR & "\Mvgetd.txt")
Call BvpEnv("PACT40","PAC7ET",Rep_USR & "\GETDETT40.txt")
Call BvpEnv("PACT40","PAC7DD",Rep_USR & "\GETDDDT40.txt")
Call RunCmdLog ("BVPACT40")
If Return = 8 Then
Call Msg_Log (Array("1027"))
End If
Call Err_Cod(Return , 0 , "PACT40")

Call Msg_Log (Array("1022" , "PACT45"))
'-----------------------------------
Call BvpEnv("PACT45","PAC7MD",Rep_USR & "\Mvgetd.txt")
Call BvpEnv("PACT45","PAC7ND",Rep_USR & "\Ndgetd.txt")
Call RunCmdLog ("BVPACT45")
Call Err_Cod(Return , 0 , "PACT45")

Call Msg_Log (Array("1022" , "PACT50"))
'--------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PAC7TD") = Rep_BASE & "\TD"
Call BvpEnv("PACT40","PAC7MD",Rep_USR & "\Mvgetd.txt")
Call BvpEnv("PACT50","PAC7ET",Rep_USR & "\GETDETT50.txt")
```

```
Call RunCmdLog ("BVPACT50")
Call Err_Cod(Return , 0 , "PACT50")

Else

Call Msg_Log (Array("1044" , Rep_BASE & "\TD"))

End If

Call Msg_Log (Array("1023"))
'--------------------------------
Call DeleteFldr(Rep_TMP)
Wscript.Quit (Return)

</script>
</job>
```

GETA - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'             - TABLES DESCRIPTION GENERATION -
'
' ------------------------------------------------------
'
' TABLE DESCRIPTION GENERATOR IS THE INTERFACE BETWEEN
' THE SPECIFICATIONS DICTIONARY AND VA PACTABLES.
' FOR FURTHER INFORMATION, REFER TO CHAPTER GENERAL
' INTRODUCTION SUBCHAPTER INTRODUCTION TO THE
' VA PACTABLES FACILITY IN THE VA PACTABLES MANUAL.
'
' GETA IF THE DICTIONARY AND VA PACTABLES ARE RUNNING
' UNDER THE SAME ENVIRONMENTS.
'
' ------------------------------------------------------
'
<job id=GETA>

<script language="VBScript">
Dim MyProc
MyProc = "GETA"
</script>

<script language="VBScript" src="INIT.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If

If  FSO.FileExists(Rep_BASE & "\TD") Then

Call Msg_Log (Array("1022" , "PACT40"))
'-------------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
```

```
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PAC7AY") = Rep_BASE & "\AY"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7TD") = Rep_BASE & "\TD"
WshEnv("PAC7MB") = Fic_Input
Call BvpEnv("PACT40","PAC7MD",Rep_USR & "\Mvgeta.txt")
Call BvpEnv("PACT40","PAC7ET",Rep_USR & "\GETAETT40.txt")
Call BvpEnv("PACT40","PAC7DD",Rep_USR & "\GETADDT40.txt")
Call RunCmdLog ("BVPACT40")
If Return = 8 Then
Call Msg_Log (Array("1027"))
End If
Call Err_Cod(Return , 0 , "PACT40")

Call Msg_Log (Array("1022" , "PACT45"))
'-----------------------------------
Call BvpEnv("PACT45","PAC7MD",Rep_USR & "\Mvgeta.txt")
Call BvpEnv("PACT45","PAC7ND",Rep_USR & "\Ndgeta.txt")
Call RunCmdLog ("BVPACT45")
Call Err_Cod(Return , 0 , "PACT45")

Else

Call Msg_Log (Array("1044" , Rep_BASE & "\TD"))

End If

Call Msg_Log (Array("1023"))
'-----------------------------
Call DeleteFldr(Rep_TMP)
Wscript.Quit (Return)

</script>
</job>
```

GETI - Initialization of Description Line

GETI - Introduction

The GETI procedure must be executed when first using Pactables files that are stored in an environment other than the VisualAge Pacbase environment. It initializes the description file in a way similar to the Pactables INTA procedure does.

GETI - User Input

An '*' line with a user code and password.

An 'I' line with initialization parameters.

Position	Length	Value	Meaning
2	1	'I'	Line code
3	36		Installation name
39	1		Language code
		'F'	French (default option)
		'E'	English
53	4	cccc	Class for Security System
57	1		Type of Security System
		'R'	RACF
		'S'	Top secret
58	2	nn	Number of lines per printed page
60	1		Type of resource controls
		' '	Definition of Security system tables resources
		'P'	Definition of resources in the Development Database
61	1		User code lock
		' '	other user code authorized
		'N'	other user code not authorized

GETI - Description of Steps

Initialization of descriptions file: PACTIN

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database Index file
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database Data file
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database Users
PAC7AR	Database dir. : AR	Input	Development Database Data file
PAC7MB	User input	Input	Parameter line
PAC7TD	Database dir. : TD	Output	Table description file
PAC7ED	User dir. : GETIEDTIN	Report	Initialization report

Code	Physical name	Type	Label
PAC7DD	User dir. : GETIDDTIN	Report	Authorization control

GETI - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'

' ------------------------------------------------------
'     - INITIALIZATION OF TABLES MANAGEMENT FILE -
'
' ------------------------------------------------------
'
' THE GETI PROCEDURE MUST BE EXECUTED WHEN FIRST USING
' VA PACTABLES FILES THAT ARE STORED IN ANOTHER
' ENVIRONMENT FROM THE PRODUCT ENVIRONMENT.
' IT INITIALIZES THE DESCRIPTION FILE IN A SIMILAR WAY
' AS THE VA PACTABLES INTA PROCEDURE DOES.
'
' ------------------------------------------------------
'
<job id=GETI>

<script language="VBScript">
Dim MyProc
MyProc = "GETI"
</script>

<script language="VBScript" src="INIT.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If

Call Msg_Log (Array("1022" , "PACTIN"))
'------------------------------
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7TD") = Rep_BASE & "\TD"
WshEnv("PAC7MB") = Fic_Input
Call BvpEnv("PACTIN","PAC7ED",Rep_USR & "\GETIEDTIN.txt")
Call BvpEnv("PACTIN","PAC7DD",Rep_USR & "\GETIDDTIN.txt")
Call RunCmdLog ("BVPACTIN")
Call Err_Cod(Return , 0 , "PACTIN")

Call Msg_Log (Array("1023"))
'------------------------------
Call DeleteFldr(Rep_TMP)
```

```
Wscript.Quit (Return)

</script>
</job>
```

SMTD-RMTD - Migration of Tables Descriptions

SMTD - Introduction

The SMTD procedure backs up the TD table-description file by transforming binary characters into their display format.

The aim of the procedure is to transfer the TD file onto different platforms while avoiding problems caused by the presence of these characters at the time of transfers.

Execution condition

None.

SMTD - Description of Steps

TD Backup (Tables description file): PTASVD

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database Index file
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database Data file
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database Users
PAC7MB	User input	Input	User transactions
PAC7TC	Save dir. : PD-new	Output	Table description backup for the migration
PAC7TD	Base dir. : TD	Input	Tables description file
PAC7DD		Report	Authorization control

SMTD - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'   - BACKUP OF TABLE DESCRIPTIONS FOR MIGRATION -
'
```

```
' ------------------------------------------------------
'
' THE SMTD PROCEDURE BACKS UP THE TD TABLE-DESCRIPTION
' FILE BY TRANSFORMING BINARY CHARACTERS INTO THEIR
' DISPLAY FORMAT.
'
' ------------------------------------------------------
'
<job id=SMTD>

<script language="VBScript">
Dim MyProc
MyProc = "SMTD"
</script>

<script language="VBScript" src="INIT.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If

Call Msg_Log (Array("1022" , "PTASVD"))
'-----------------------------------
WshEnv("PAC7TC") = Rep_SAVE & "\PD-new"
Call BvpEnv("PTASVD","PAC7TD",Rep_BASE & "\TD")

WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7MB") = Fic_Input
Call BvpEnv("PTASVD","PAC7DD",Rep_USR & "\SMTDDDSVD.txt")

Call RunCmdLog ("BVPTASVD")
Call Err_Cod(Return , 0 , "PTASVD")

Call Msg_Log (Array("1022" , "BACKUP"))
'-----------------------------------
Call Turnover(Rep_SAVE & "\PD")


Call Msg_Log (Array("1023"))
'------------------------------
Call DeleteFldr (Rep_TMP)
Wscript.Quit (Return)

</script>
</job>
```

RMTD - Introduction

The Restoration of Table Descriptions procedure (RMTD) restores the TD file of Table Descriptions from its TC sequential backup produced by the SMTD procedure.

This procedure does not require any specific execution condition.

RMTD - Description of Steps

TD File Restoration: PTARSD

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database Index file
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database Data file
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database Users
PAC7MB	User input	Input	User transactions
PAC7TD	Save dir. : PD	Output	Tables Description file
PAC7TC	Database dir. : TD	Input	Tables Description sequential file
PAC7DD		Report	Authorization control

RMTD - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'            - RESTORATION OF TABLE DESCRIPTIONS -
'
' ------------------------------------------------------
'
' THE RESTORATION OF TABLE DESCRIPTIONS PROCEDURE
' (RMTD)    RESTORES THE TD FILE OF
' TABLE DESCRIPTIONS FROM ITS TC SEQUENTIAL BACKUP
' PRODUCED BY THE SMTD PROCEDURE.
'
' ------------------------------------------------------
'
<job id=RMTD>

<script language="VBScript">
Dim MyProc
MyProc = "RMTD"
</script>

<script language="VBScript" src="INIT.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If
```

```
Call Msg_Log (Array("1022" , "PTARSD"))
'-----------------------------------
WshEnv("PAC7TC") = Rep_SAVE & "\PD"
If Not FSO.FileExists(WshEnv("PAC7TC")) Then
  Call Msg_Log (Array("1004" , "PAC7TC"))
  Msg = Nls_Lib
  EndJob (1)
  Wscript.Quit (1)
End If
WshEnv("PAC7TD") = Rep_BASE & "\TD"

WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7MB") = Fic_Input
Call BvpEnv("PTARSD","PAC7DD",Rep_USR & "\RMTDDDRSD.txt")

Call RunCmdLog ("BVPTARSD")
Call Err_Cod(Return , 0 , "PTARSD")

Call Msg_Log (Array("1022" , "BACKUP"))
'-----------------------------------
Call Turnover(Rep_SAVE & "\PD")


Call Msg_Log (Array("1023"))
'---------------------------
Call DeleteFldr (Rep_TMP)
Wscript.Quit (Return)

</script>
</job>
```

Chapter 7. Pac/Impact

Foreword

Note: Pac/Impact users may also refer to the 'Pac/Impact for VisualAge Pacbase' manual.

Warning

An Impact analysis requires a very large amount of machine-time. It is therefore recommended to limit the scope of the analysis.

You can limit your analysis to two distinct levels. You can also combine two levels, to define a more precise analysis domain.

- The UXSR procedure, documented in 'The Administrator's Procedures' manual, allows you to create a new image of the Development Database, by zooming on a given sub-network (the session extraction is also available). This creates a new Database which is a subset (restructured and/or renamed) of the initial Database. The analysis is then performed on this subset.

  Furthermore, the REOR procedure (which must always be run after a UXSR) allows you to cancel instances which are not relevant to the analysis.
- You may also choose to limit your analysis to some instances of the Program, Screen or Database Block entities. Additional selection options are available to this effect.

  This analysis limitation is performed by the INFP utility, documented in this manual.
- The procedures in this Function do not impact the Database files. However, it is recommended to close the on-line files for better performance.

INFP - FP File Initialization (Impact Analysis)

INFP - Introduction

This procedure allows to specify the entities which are to be analyzed and thus to narrow the scope of the impact analysis.

For the FP file to be updated by INFP, you must re-enter, in the procedure's input, all the lines already entered. You always start with an empty file, i.e. a file which contains no particular selection.

Result

The procedure outputs a file which contains the entities selected for the analysis (FP).

INFP - User Input

A '*' line with the user code and password.

Other input is optional, knowing that if no input is provided, all the entities of all entity types will be searched for in the impact analysis.

If you request all the existing entities of a given entity type (code = ******), you cannot indicate any specific entities for this type.

If you specify a type in an input line (whether or not you specify an entity for this type), you must also specify, on additional input lines, all the other types to be analyzed by the procedure.

Position	Length	Value	Meaning
1	3		Entity type Possible values are:
		'B '	Database Blocks
		'F '	Meta-Entities
		'O '	Screens
		'P '	Programs
		'T '	Texts
		'V '	Documents
		$nn	User Entities of 'nn' type code.
		'$**'	All UEs
4	6		Entity code (generic selection through code ******) (This code may not exist in the Database)

INFP - Description of Steps

Check on transactions and FP update: PAN205

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database Index file
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database Data file
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database Users

Code	Physical name	Type	Label
PAC7AR	Database dir. : AR	Input	Development Database Data file
PAC7MB	User input	Input	User input
PAC7FP	Base dir. : FP	Output	Entities in production
PAC7IP	User dir. : INFPIP205	Report	Validation report

Return codes:

- 0 : OK.
- 12 : System error.

INFP - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'     - IMPACT ANALYSIS:  INITIALIZATION OF "FP" FILE -
'
' ------------------------------------------------------
'
' THE INFP PROCEDURE INITIALIZES THE FP FILE. IT ALLOWS
' TO SPECIFY THE ENTITIES WHICH ARE TO BE ANALYZED AND
' THUS TO NARROW THE SCOPE OF THE IMPACT ANALYSIS TO SOME
' (OR ALL) OCCURRENCES OF THE ENTITIES.
'
' ------------------------------------------------------
'
<job id=INFP>

<script language="VBScript">
Dim MyProc
MyProc = "INFP"
</script>

<script language="VBScript" src="INIT.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If

Call Msg_Log (Array("1022" , "PAN205"))
'--------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7FP") = Rep_BASE & "\FP"
WshEnv("PAC7MB") = Fic_Input
```

```
Call BvpEnv("PAN205","PAC7IP",Rep_USR & "\INFPIP205.txt")
Call RunCmdLog ("BVPAN205")
If Return = 12 Then
Call Msg_Log (Array("1026" , "PAN205"))
End If
Call Err_Cod(Return , 0 , "PAN205")

Call Msg_Log (Array("1023"))
'------------------------------
Call DeleteFldr (Rep_TMP)
Wscript.Quit (Return)

</script>
</job>
```

ISEP - Selection of Entry Points

ISEP - Introduction

The ISEP procedure is designed to select the entry points -- Elements and/or character strings -- which will be used as criteria by the impact analysis (IANA procedure).

The identification line of the selection context (* line) is required. It allows you to specify the session and the sub-network (view Z1) from which the selection will be made.

Elements and character strings are considered as entry points when they meet the selection criteria entered in ISEP user input lines.

Three types of criteria may be used (see below) and at least one selection criterion is required, knowing that no particular criterion type is required.

A selection may combine several types of criteria, and several command lines for each type.

- The E-type line allows you to extract Elements by selecting a code (generic code authorized) and/or one or several format(s).
- The S-type line allows you to extract character strings by selecting a code (generic code authorized) and/or one or several format(s).
- The W-type line allows you to select Elements via a keyword. You may also indicate the keyword type, Element formats and code.

Execution conditions

None.

Abnormal execution

Whatever the cause of the abend, the procedure can be restarted as it is, once the problem has been solved.

Result

Output of the ISEP procedure is two files which are to be used in the IANA procedure:

- 'FH' file which contains the selected entry points,
- 'FR' file which contains the entry points to be purged.

ISEP - User Input

Only one '*' line (required, located at the beginning of the stream):

Position	Length	Value	Meaning
2	1	'*'	Line code
3	8	uuuuuuuu	User code
11	8	pppppppp	Password
19	3	bbb	Code of the highest library in the sub-network
22	4	ssss	Session number (blank if current session)
26	1		Session status (' ' or 'T')
69	3	iii	Code of the lowest Library in the sub-network (optional)

One E-type line: Selection of Elements (optional):

Position	Length	Value	Meaning
2	1	'E'	Line code
3	6		Element code (generic code possible with the '*' character, at beginning or end of code: ***XXX or XXX***, or with the '?' character followed by the string to be included in the code (?XXX))
9	10		Element input format
19	10		Element internal format
29	1		Internal usage (default: D)
30	27		Element output format
57	1	'N'	Child Elements not impacted
		' '	Child Elements impacted

One S-type line: Selection of character strings (optional)

Position	Length	Value	Meaning
2	1	'S'	Line code
3	30		String code (generic code possible with the '*' character anywhere in the code), or ?xx where xx is a string located anywhere in the sequence of char.
33	10		Internal format of the string
43	1		Internal usage (Default: D)

One W-type line: Selection on keyword (optional)

Position	Length	Value	Meaning
2	1	'W'	Line code
3	1		Keyword type (implicit 'L', explicit 'M', or both ' ')
4	13		Keyword code (no generic code)
17	10		Element input format
27	10		Element internal format
37	1		Internal usage (Default: D)
38	27		Element output format
65	6		Element code (generic code possible with the '*' character anywhere in the code)
71	1	'N'	Child Elements not impacted
		' '	Child Elements impacted

ISEP - Description of Steps

Selection of entry points: PAN210

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database Index file
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database Data file
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database Users

Code	Physical name	Type	Label
PAC7AR	Database dir. : AR	Input	Development Database Data file
PAC7AN	Database dir. : AN	Input	Development Database Index file
PAC7FP	Database dir. : FP	Input	File of entities to be analyzed
PAC7MB	User input	Input	User input
PAC7FH	Tmp dir. : WFH	Output	Selected entry points
PAC7IE	User dir. : ISEPIE210	Report	Validation report

Return Codes :

- 0 : OK.
- 12 : System error

Removal of duplicate entry points: PAN215

Code	Physical name	Type	Label
PAC7FH	Tmp dir. : WFH	Input	Selected entry points
PAC7HF	Database dir. : FH-new	Output	Sorted selected entry points
PAC7FR	Database dir. : FR-new	Output	Reduced entry points to be purged

.Return codes:

- 0 : OK.
- 12 : System error.

File Rotation

The NEW file is created. To continue the impact analysis, a file rotation must be made to obtain the current file.

FH-NEW --(rotation)--> FH , the old FH file becomes FH-1.

FR-NEW --(rotation)--> FR , the old FR file becomes FR-1.

ISEP - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
```

```
'       - IMPACT ANALYSIS : SELECTION OF ENTRY POINTS -
'
' -----------------------------------------------------
'
' THE ISEP PROCEDURE IS DESIGNED TO SELECT THE ENTRY
' POINTS -- DATA ELEMENTS AND/OR CHARACTER STRINGS --
' WHICH WILL BE USED AS CRITERIA BY THE IMPACT
' ANALYSIS (IANA PROCEDURE).
'
' -----------------------------------------------------
'
<job id=ISEP>

<script language="VBScript">
Dim MyProc
MyProc = "ISEP"
</script>

<script language="VBScript" src="INIT.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If

Call Msg_Log (Array("1022" , "PAN210"))
'-------------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7FP") = Rep_BASE & "\FP"
WshEnv("PAC7MB") = Fic_Input
Call BvpEnv("PAN210","PAC7FH",Rep_TMP & "\WFH.tmp")
Call BvpEnv("PAN210","PAC7IE",Rep_USR & "\ISEPIE210.txt")
Call RunCmdLog ("BVPAN210")
If Return = 12 Then
Call Msg_Log (Array("1026" , "PAN210"))
End If
Call Err_Cod(Return , 0 , "PAN210")

Call Msg_Log (Array("1022" , "PAN215"))
'---------------------------------
Call BvpEnv("PAN215","PAC7FH",Rep_TMP & "\WFH.tmp")
WshEnv("PAC7HF") = Rep_BASE & "\FH-new"
WshEnv("PAC7FR") = Rep_BASE & "\FR-new"
Call RunCmdLog ("BVPAN215")
If Return = 12 Then
Call Msg_Log (Array("1026" , "PAN215"))
End If
Call Err_Cod(Return , 0 , "PAN215")

Call Msg_Log (Array("1022" , "BACKUP"))
'-------------------------------
```

```
Call Turnover(Rep_BASE & "\FH")
Call Turnover(Rep_BASE & "\FR")

Call Msg_Log (Array("1024"))
'------------------------------
Call DeleteFldr(Rep_TMP)

Call Msg_Log (Array("1023"))
'----------------------------
Wscript.Quit (Return)

</script>
</job>
```

ISOS - Selection of Strings and Operators

ISOS - Introduction

ISOS is a complement to the ISEP procedure. Its purpose is to select the following items:

- VA Pac-processed dates, such as DATOR and DAT8, that will be used as entry points to perform the impact analysis from the first iteration (IANA procedure),
- Character-strings, without considering them as entry points (such as ORDER BY). For the strings which provide entry points, see the description of the 'S'-type line in the ISEP procedure's USER INPUT section,
- Operators used in procedural code (-P) lines, such as ADT. Some of these operators trigger the generation of date-type entry points (such as DATOR for ADT),
- Lines that use constant values, either defined (VALUE), moved (MOVE), or conditioned ('IF').

The restoration of the entities which use these operators and character-strings can be executed on request (IPIA procedure).

Narrowing the scope of selection

For better performance, it is advisable to narrow the scope of the selection. This can be done at two different levels, and should always be done before running the procedure.

- Via the UXSR procedure, documented in 'The Administrator's Procedures' manual, you can create another Development Database. The new Database is a subset (restructured and/or renamed) of the initial Database. The analysis will be performed on this subset.
- Via the INFP utility, documented in this manual: FP File Initialization (Impact Analysis)', you can decide to restrict the scope of the selection to

entities of a particular type or types, or to particular entities of a given type. Further selection options are also available.

The selection context's identification line (*-line) is required. It allows you to specify, besides the session, the Library from which you want to build the sub-network that will be analyzed (view Z1).

Three types of selection may be used (see below). At least one type of selection is required, no particular type being requested.

The selection may include more than one type of selection, and more than one command line for each type.

- The 'D'-type line allows you to request the extraction of date-type Elements handled by VisualAge Pacbase.

  The maximum number of 'D'-lines is 40.
- The 'C'-type line allows you to extract character-strings that are likely to include one or more blanks. In this case, the separator must be specified, and the number of blanks is significant. These strings are not entry points.

  The maximum number of 'C'-lines is 50 characters for each one of the three search domains.
- The 'O'-type line allows you to select operators processed in -P lines.

  The maximum number of 'O'-lines is 50.

Execution conditions

None.

Abnormal execution

Whatever the cause of an abnormal ending, the procedure may be restarted as it is after correction of the problem.

Result

The output of the ISOS procedure is:

- an 'FH' file which contains the selected entry points, to be used by the IANA procedure,
- an 'FR' file which contains the entry points to be purged, to be used by the IANA procedure,
- an 'FO' file, which contains the analysis results, to be used by the IANA or IPIA procedure.

ISOS - User Input

Only one '*'-line (required, located at the beginning of the stream):

Position	Length	Value	Meaning
2	1	'*'	Line code
3	8	uuuuuuuu	User code
11	8	pppppppp	Password
19	3	bbb	Code of the highest Library in the sub-network
22	4	ssss	Session number (blank if current session)
26	1		Session status (' ' or 'T')
28	1	'F' or 'E'	Language code if different from that of the site (bilingual sites only)
69	3	iii	Code of the lowest Library in the sub-network (optional)

One 'D'-line for the selection of generated dates (optional):

Position	Length	Value	Meaning
2	1	'D'	Line code
3	9		Code of generated date Element to be extracted (which must be recognized by the system)

One 'O'-line for the selection of operators (optional):

Position	Length	Value	Meaning
2	1	'O'	Line code
3	3		Code of searched operator (which must be recognized by the system)

One 'C'-line for the selection of character strings (optional):

Position	Length	Value	Meaning
2	1	'C'	Line code
3	1		End-of-string separator (Required if the string contains at least one blank)

Position	Length	Value	Meaning
4	31		Code of searched string. (Must be ended by the separator if a separator is specified)
35	1		Where the string is to be searched:
		'D'	Search in the Definition part (-W of Programs and/or Screens, and -9 of programs)
		'T'	Search in Procedural Code part (-P of programs and/or screens, -8, -9, -SC of programs, -CE and -CS of screens)
		'R'	Search in the Report specific part: .Category condition and Structure .Source Element code (Struct.)
		' '	Search in the three above mentioned parts

One 'V'-line for the selection of constant values (optional):

Position	Length	Value	Meaning
2	1	'V'	Line code
3	1		Beginning-of-value separator Required (either ' or ")
4	31		Code of searched value Required, ending with the separator (either ' or ")
35	1		Where the constant is to be searched
		'D'	Search in the Definition part (-W of Programs and/or Screens, and -9 of Programs)
		'T'	Search in the Procedural Code part (-P of Programs and/or Screens, -8, -9, -SC of Programs, -CE and -CS of Screens)
		'R'	Search in the Report specific part: .Category condition and Structure .Source Element code (Struct.)
		' '	Search in the three above mentioned parts

ISOS - Description of Steps

Selection of strings and operators: PAN212

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administrator Database Index file
PACGGR	Admin Database - Base dir. : AR	Input	Administrator Database Data file
PACGGU	Admin Database - Base dir. : GU	Input	Administrator Database Users
PAC7AR	Database dir. : AR	Input	Development Database Data file
PAC7AN	Database dir. : AN	Input	Development Database Index file
PAC7FP	Database dir. : FP	Input	Entities to be analyzed
PAC7MB	User input	Input	User input
PAC7FH	Tmp dir. : WFH	Output	Selected entry points (length=160)
PAC7MF	Tmp dir. : WFO	Output	Impact analysis result (length=266)
PAC7IE	User dir. : ISOSIE212	Report	Validation control

.Return Codes :

- 0: OK
- 12: System error

Deletion of duplicate entry points: PAN215

Code	Physical name	Type	Label
PAC7FH	Tmp dir. : WFH	Input	Selected entry points
PAC7HF	Database dir. : FH-new	Output	Sorted selected entry points
PAC7FR	Database dir. : FR-new	Output	Reduced entry points to be purged

Return codes :

- 0 : OK
- 12 : System error

Update of impact analysis results: PAN260

Code	Physical name	Type	Label
PAC7MF	Tmp dir. : WFO	Input	Impact analysis result (for that iteration)
PAC7OF	Database dir. : FO	Input	Results from preceding analysis
PAC7FO	Database dir. : FO-new	Output	Sorted impact-analysis results

Return codes:

- 0 : OK.
- 12 : System error.

File Rotation

The NEW file is created. To continue the impact analysis, a file rotation must be made to obtain the current file.

FH.NEW --(rotation)--> FH , the old FH file becomes FH-1.

FR.NEW --(rotation)--> FR , the old FR file becomes FR-1.

FO.NEW --(rotation)--> FO , the old FO file becomes FO-1.

ISOS - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
' - IMPACT ANALYSIS: SELECTION OF STRINGS AND OPERATORS
'
' ------------------------------------------------------
'
' ISOS IS A COMPLEMENT TO THE ISEP PROCEDURE.
' FOR BETTER PERFORMANCE, IT IS ADVISABLE TO NARROW THE
' SCOPE OF THE SELECTION. THIS CAN BE DONE
' AT TWO DIFFERENT LEVELS, AND SHOULD ALWAYS
' BE DONE BEFORE RUNNING THE PROCEDURE.
'
' ------------------------------------------------------
'
<job id=ISOS>

<script language="VBScript">
Dim MyProc
MyProc = "ISOS"
</script>
```

```
<script language="VBScript" src="INIT.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If

Call Msg_Log (Array("1022" , "PAN212"))
'-------------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7FP") = Rep_BASE & "\FP"
WshEnv("PAC7MB") = Fic_Input
Call BvpEnv("PAN212","PAC7FH",Rep_TMP & "\WFH.tmp")
Call BvpEnv("PAN212","PAC7MF",Rep_TMP & "\WFO.tmp")
Call BvpEnv("PAN212","PAC7IE",Rep_USR & "\ISOSIE212.txt")
Call RunCmdLog ("BVPAN212")
If Return = 12 Then
Call Msg_Log (Array("1026" , "PAN212"))
End If
Call Err_Cod(Return , 0 , "PAN212")

Call Msg_Log (Array("1022" , "PAN215"))
'-------------------------------------
Call BvpEnv("PAN215","PAC7FH",Rep_TMP & "\WFH.tmp")
WshEnv("PAC7HF") = Rep_BASE & "\FH-new"
WshEnv("PAC7FR") = Rep_BASE & "\FR-new"
Call RunCmdLog ("BVPAN215")
If Return = 12 Then
Call Msg_Log (Array("1026" , "PAN215"))
End If
Call Err_Cod(Return , 0 , "PAN215")

Call Msg_Log (Array("1022" , "PAN260"))
'-------------------------------------
Call BvpEnv("PAN260","PAC7MF",Rep_TMP & "\WFO.tmp")
WshEnv("PAC7OF") = Rep_BASE & "\FO"
WshEnv("PAC7FO") = Rep_BASE & "\FO-new"
Call RunCmdLog ("BVPAN260")
If Return = 12 Then
Call Msg_Log (Array("1026" , "PAN260"))
End If
Call Err_Cod(Return , 0 , "PAN260")


Call Msg_Log (Array("1022" , "BACKUP"))
'--------------------------------
Call Turnover(Rep_BASE & "\FH")
Call Turnover(Rep_BASE & "\FO")
Call Turnover(Rep_BASE & "\FR")
```

```
Call Msg_Log (Array("1024"))
'--------------------------------
Call DeleteFldr (Rep_TMP)

Call Msg_Log (Array("1023"))
'----------------------------
Wscript.Quit (Return)

</script>
</job>
```

IMFH - Merge of FH Files - Creation of FH and FR

IMFH - Introduction

The IMFH procedure allows you to merge two or more FH files (selected entry points) so as to:

- Have only one FH file, after eliminating possible duplicates;
- Obtain an FR file (entry points to be purged) in phase with the FH file created.

This procedure should be used when you want to merge the FH file produced by the ISEP procedure with that issued by the ISOS procedure.

A subsidiary use of this procedure is to recreate the FR file from an FH file.

Result

The ISEP procedure outputs two files which are to be used by the IANA procedure:

- an 'FH' file which contains the selected entry points,
- an 'FR' file which contains the entry points to be purged.

IMFH - Description of Steps

Deletion of duplicate entry points: PAN215

Code	Physical name	Type	Label
PAC7FH	Database dir. : FH	Input	Selected entry points to be merged
PAC7HF	Database dir. : FH-new	Output	Sorted selected entry points
PAC7FR	Database dir. : FR-new	Output	Reduced entry points to be purged

Return codes:

- 0 : OK.

- 12 : System error.

IMFH - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
' - IMPACT ANALYSIS: MERGE FH FILES AND CREATION FR FILE
'
' ------------------------------------------------------
'
' THIS PROCEDURE SHOULD BE USED WHEN YOU WANT TO MERGE
' THE FH FILE PRODUCED BY THE ISEP PROCEDURE WITH THAT
' ISSUED BY THE ISOS PROCEDURE.
'
'
' ------------------------------------------------------
'
<job id=IMFH>

<script language="VBScript">
Dim MyProc
MyProc = "IMFH"
</script>

<script language="VBScript" src="INIT.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If

Call Msg_Log (Array("1022" , "PAN215"))
'---------------------------------------

If  FSO.FileExists(Rep_BASE & "\FH-1") Then
 Set MyFile = fso.GetFile(Rep_BASE & "\FH-1")
 MyFile.Copy (Rep_TMP & "\WFH.tmp")
else
 Set MyFile = FSO.CreateTextFile(Rep_TMP & "\WFH.tmp")
 MyFile.Close
end if
If  FSO.FileExists(Rep_BASE & "\FH") Then
  Call CopMFil (Rep_TMP  & "\WFH.tmp" ,Rep_BASE & "\FH" , Rep_TMP  & "\WFH.tmp")
end if

Call BvpEnv("PAN215","PAC7FH",Rep_TMP & "\WFH.tmp")

WshEnv("PAC7HF") = Rep_BASE & "\FH-new"
WshEnv("PAC7FR") = Rep_BASE & "\FR-new"
Call RunCmdLog ("BVPAN215")
If Return = 12 Then
Call Msg_Log (Array("1026" , "PAN215"))
End If
Call Err_Cod(Return , 0 , "PAN215")
```

```
Call Msg_Log (Array("1022" , "BACKUP"))
'-------------------------------
Call Turnover(Rep_BASE & "\FH")
Call Turnover(Rep_BASE & "\FR")

Call Msg_Log (Array("1024"))
'-------------------------------
Call DeleteFldr (Rep_TMP)

Call Msg_Log (Array("1023"))
'----------------------------
Wscript.Quit (Return)

</script>
</job>
```

INFQ - FQ File Reinitialization (Impact Analysis)

INFQ - Introduction

The INFQ procedure reinitializes the FQ file, which accumulates all the search criteria that have already been impacted by the analysis. This accumulation prevents these criteria from being analyzed again in future analyses.

This action should be performed before a new impact analysis either because the entry points have changed or because the analysis context has changed.

However, it must not be used between two iterations of the same impact analysis.

User input

A '*' line with the user code and password.

Result

The procedure outputs a reinitialized file of search criteria (FQ).

INFQ - Description of Steps

FQ file Reinitialization : PAN200

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database Index file
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database Data file

Code	Physical name	Type	Label
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database Users
PAC7AR	Database dir. : AR	Input	Development Database Data file
PAC7MB	User input	Input	User input
PAC7FQ	Database dir. : FQ-new	Output	Impacted criteria reinitialized sequential file
PAC7DD	User dir. : INFQDD200	Report	Error report

INFQ - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'     - IMPACT ANALYSIS:  INITIALIZATION OF "FQ" FILE -
'
' ------------------------------------------------------
'
' THIS ACTION SHOULD BE PERFORMED BEFORE A NEW IMPACT
' ANALYSIS EITHER BECAUSE THE ENTRY POINTS HAVE CHANGED
' OR BECAUSE THE ANALYSIS CONTEXT HAS CHANGED.
'
' ------------------------------------------------------
'
<job id=INFQ>

<script language="VBScript">
Dim MyProc
MyProc = "INFQ"
</script>

<script language="VBScript" src="INIT.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If

Call Msg_Log (Array("1022" , "PAN200"))
'-------------------------------------
WshEnv("PAC7MB") = Fic_Input
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
Call BvpEnv("PAN200","PAC7DD",Rep_USR & "\INFQDD200.txt")
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7FQ") = Rep_BASE & "\FQ-new"
Call RunCmdLog ("BVPAN200")
If Return = 12 Then
```

```
Call Msg_Log (Array("1026" , "PAN200"))
End If
Call Err_Cod(Return , 0 , "PAN200")

Call Msg_Log (Array("1022" , "BACKUP"))
'-------------------------------------
Call Turnover(Rep_BASE & "\FQ")

Call Msg_Log (Array("1023"))
'-----------------------------
Call DeleteFldr(Rep_TMP)
Wscript.Quit (Return)

</script>
</job>
```

IGRA - Breaking down of Group Fields

IGRA - Introduction

The IGRA procedure breaks down group fields into Elementary Fields. These group fields can be:

- Entry points detected by the ISEP procedure.
- Impact search criteria obtained by running the IANA procedure.

The IGRA procedure is optional and does not generate any impact search criterion.

Before running the IGRA procedure, you may purge:

- Entry points --after executing the ISEP procedure.
- Impact search criteria --after executing the IANA procedure which precedes.

In both cases, deletions are made in the FR file (via an editor) by inhibiting them (value 'E' in the action code of the corresponding lines), in order to save them for future executions of IANA.

It is not necessary to eliminate non-Group fields since they will simply be ignored by the procedure.

The notions of 'level' and 'iterations' are not relevant for the IGRA procedure.

Entry points (first iteration) or impact search criteria (further iterations) are printed once the purged criteria have been taken into account. This printout sorts criteria into 'accepted' and 'rejected' criteria.

The impact results file may either be empty or contain the results of other IANA, ISOS, or IGRA executions, either in the same execution context or in

different contexts. This allows you to compound the results of all iterations of the impact analysis for one or several contexts.

Restitution of all the information for a given context may be customized (parameter setting) when printing with the IPIA procedure.

The file of Entities to be analyzed (FP) is used as input to this procedure. It contains a list of Entities or Entity Types which should be analyzed. If no user input is entered in this file before its initialization by the INFP procedure, all analyzable Entities will be analyzed.

Entities to be analyzed are specified as follows: 3-character Type, and 6-character code (****** being the Entity generic code).

Execution conditions

None, except that the FH file (entry points or impact search criteria) must exist and must not be empty.

Abnormal execution

Whatever the cause of the abnormal ending, the procedure may be restarted as it is after correcting the problem. However, the status of generation files (FH, FR, and FO) should be checked.

Result

The procedure outputs a file which contains the analysis results (FO) to be used in the IPIA procedure.

User input

One '*' line with user code and password.

IGRA - Description of Steps

Recognition of purged criteria: PAN230

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database Index file
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database Data file
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database Users

Code	Physical name	Type	Label
PAC7AR	Database dir. : AR	Input	Development Database Data file
PAC7MB	User input	Input	User input
PAC7FH	Database dir. : FH	Input	Search criteria file
PAC7FR	Database dir. : FR	Input	Reduced file of purged criteria
PAC7HF	Tmp dir. : WHF	Output	Search criteria file (length=160)
PAC7DD	User dir. : IGRADD230	Report	Error file

Return codes :

- 0 : OK
- 12 : System error

Printing of entry points: PAN220

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7HF	Tmp dir. : WHF	Input	Sorted criteria file
PAC7IL	User dir. : IGRAIL220	Report	List of accepted/rejected criteria

Return codes :

- 0 : OK
- 12 : System error

Breaking down of group fields: PAN255

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database Index file
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database Data file
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database Users
PAC7AR	Database dir. : AR	Input	Development Database Data file

Code	Physical name	Type	Label
PAC7AN	Database dir. : AN	Input	Development Database Data index file
PAC7FP	Database dir. : FP	Input	Entities to analyze
PAC7FH	Tmp dir. : WHF	Input	Impacted criteria
PAC7MF	Tmp dir. : WFO	Output	Impact analysis results (length=266)

Return Codes :

- 0 : OK
- 12 : System error

Update of impact analysis results: PAN260

Code	Physical name	Type	Label
PAC7MF	Tmp dir. : WFO	Input	Impact analysis result (by level)
PAC7OF	Database dir. : FO	Input	Results of previous analysis
PAC7FO	Base dir. : FO-new	Output	Sorted results of the impact analysis

Return codes:

- 0 : OK
- 12 : System error

IGRA - Execution Script

```
' -------------------------------------------------------
'      VISUALAGE PACBASE
'
' -------------------------------------------------------
'            - IMPACT ANALYSIS:  PRINT OF "FQ" FILE  -
'
' -------------------------------------------------------
'
' THE IGRA PROCEDURE BREAKS DOWN GROUP FIELDS
' INTO ELEMENTARY FIELDS:
' 1. ENTRY POINTS DETECTED BY THE ISEP PROCEDURE, IF
' THEY ARE OF THE GROUP TYPE.
' 2. IMPACT SEARCH CRITERIA OBTAINED BY RUNNING THE IANA
' PROCEDURE, IF THEY ARE OF THE GROUP TYPE.
'
' -------------------------------------------------------
'
<job id=IGRA>
```

```
<script language="VBScript">
Dim MyProc
MyProc = "IGRA"
</script>

<script language="VBScript" src="INIT.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If

Call Msg_Log (Array("1022" , "PAN230"))
'-------------------------------------
WshEnv("PAC7MB") = Fic_Input
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
Call BvpEnv("PAN230","PAC7DD",Rep_USR & "\IGRADD230.txt")
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7FH") = Rep_BASE & "\FH"
WshEnv("PAC7FR") = Rep_BASE & "\FR"
Call BvpEnv("PAN230","PAC7HF",Rep_TMP & "\WHF.tmp")
Call RunCmdLog ("BVPAN230")
If Return = 12 Then
Call Msg_Log (Array("1026" , "PAN230"))
End If
Call Err_Cod(Return , 0 , "PAN230")

Call Msg_Log (Array("1022" , "PAN220"))
'-------------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
Call BvpEnv("PAN220","PAC7HF",Rep_TMP & "\WHF.tmp")
Call BvpEnv("PAN220","PAC7IL",Rep_USR & "\IGRAIL220.txt")
Call RunCmdLog ("BVPAN220")
If Return = 12 Then
Call Msg_Log (Array("1026" , "PAN220"))
End If
Call Err_Cod(Return , 0 , "PAN220")

Call Msg_Log (Array("1022" , "PAN255"))
'-------------------------------------
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7FP") = Rep_BASE & "\FP"
Call BvpEnv("PAN255","PAC7MF",Rep_TMP & "\WFO.tmp")
Call BvpEnv("PAN255","PAC7FH",Rep_TMP & "\WHF.tmp")
Call RunCmdLog ("BVPAN255")
If Return = 12 Then
Call Msg_Log (Array("1026" , "PAN255"))
End If
```

```
Call Err_Cod(Return , 0 , "PAN255")

Call Msg_Log (Array("1022" , "PAN260"))
'-------------------------------------
Call BvpEnv("PAN260","PAC7MF",Rep_TMP & "\WFO.tmp")
WshEnv("PAC7OF") = Rep_BASE & "\FO"
WshEnv("PAC7FO") = Rep_BASE & "\FO-new"
Call RunCmdLog ("BVPAN260")
If Return = 12 Then
Call Msg_Log (Array("1026" , "PAN260"))
End If
Call Err_Cod(Return , 0 , "PAN260")

Call Msg_Log (Array("1022" , "BACKUP"))
'-------------------------------------
Call Turnover(Rep_BASE & "\FO")

Call Msg_Log (Array("1024"))
'--------------------------
Call DeleteFldr (Rep_TMP)

Call Msg_Log (Array("1023"))
'-----------------------------
Wscript.Quit (Return)

</script>
</job>
```

IANA - Impact Search Criteria

IANA - Introduction

The IANA procedure is used to search impacted Data Elements and character-strings according to:

- The entry points provided by the ISEP procedure when IANA is run for the first time,
- The impact search criteria produced by a preceding execution of IANA.

IANA is therefore an iterative process, which runs until no more impact search criteria are found.

Prior to an IANA execution, you can inhibit:

- Entry points, after the execution of the ISEP procedure,
- Impact search criteria, after a preceding execution of the IANA procedure.

In both cases, deletions are made in the FR file, (under an editor) either by physical deletion, or by inhibition (value 'E' in the action code of the corresponding lines).

The entry points (first iteration) or impact search criteria (further iterations) are printed once the purged criteria have been taken into account. This printout sorts criteria into 'accepted' and 'rejected' criteria. The file which contains the already impacted criteria (FQ) may be reinitialized if you do not need to save them.

However, it is recommended to reinitialize this file before the first execution of IANA which follows a new execution of ISEP. To reinitialize the FQ file, run the INFQ procedure.

The impact analysis file may either be empty or contain the results of different execution contexts. It allows to compound the results of all iterations of the impact analysis for a given context.

The FP file used as input for the analysis procedures, contains the list of the entities or entity types to be analyzed. If no user input is entered in this file before it is initialized by the INFP procedure, all analyzable entities will be analyzed.

Entities which are to be analyzed are specified in the FP file via the following coding: type coded on 3 characters, entity coded on 6 characters (****** being the generic entity code).

Execution conditions

The FH file -- entry points or impact search criteria -- must exist and must not be empty.

Abnormal execution

Whatever the cause of the abend, you can run the procedure again as it is, after the problem has been solved.

However, the status of the FH, FR, and FO generation files should be checked.

User input

One '*' line with user code and password.

This procedure is iterative as long as the FH file (impact search criteria) is not empty (return code set to value 4 if empty, and to value 0 otherwise).

Result

This procedure outputs a file which contains the analysis results (FO) to be used in the IPIA procedure.

IANA - Description of Steps

Recognition of criteria after the purge: PAN230

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database Index file
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database Data file
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database Users
PAC7AR	Database dir. : AR	Input	Development Database Data file
PAC7MB	User input	Input	User input
PAC7FH	Base dir. : FH	Input	Search criteria file
PAC7FR	Database dir. : FR	Input	Search criteria after purge (reduced file)
PAC7HF	Tmp dir. : WHF	Output	Search criteria file (length=160)
PAC7DD	User dir. : IANADD230	Report	Error report

Return codes :

- 0 : OK
- 12 : System error

Printing of entry points: PAN220

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7HF	Tmp dir. : WHF	Input	Sorted criteria
PAC7IL	User dir. : IANAIL220	Report	List of accepted / rejected criteria

Return codes :

- 0 : OK
- 12 : System error

Impact analysis: PAN250

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database Index file
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database Data file
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database Users
PAC7AR	Database dir. : AR	Input	Development Database Data file
PAC7AN	Database dir. : AN	Input	Development Database Index file
PAC7AY	Database dir. : AY	Input	Development Database extension data
PAC7FP	Database dir. : FP	Input	File of entities to be analyzed
PAC7FH	Tmp dir. : WHF	Input	Impacted criteria
PAC7FQ	Tmp dir. : WFQ	Input / Output	Impacted criteria already processed
PAC7HF	Tmp dir. : WFH	Output	New impacted criteria (length = 160)
PAC7MF	Tmp dir. : WFO	Output	Impact analysis results (length = 266)

Return codes:

- 0 : OK
- 4 : OK. Iteration ended.
- 12 : System error

Update of impact analysis results: PAN260

Code	Physical name	Type	Label
PAC7MF	Tmp dir. : WFO	Input	Impact analysis results (level)
PAC7OF	Database dir. : FO	Input	Results of previous analysis
PAC7FO	Database dir. : FO-new	Output	Sorted results of impact analysis

Return codes:

- 0 : OK
- 12 : System error

Removal of duplicate entry points: PAN215

Code	Physical name	Type	Label
PAC7FH	Tmp dir. : WFH	Input	Selected entry points
PAC7HF	Database dir. : FH-new	Output	Sorted selected entry points
PAC7FR	Database dir. : FR-new	Output	Reduced entry points to be purged

Return codes:

- 0 : OK
- 12 : System error

IANA - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'                  - IMPACT ANALYSIS -
'
' ------------------------------------------------------
'
' THE IANA PROCEDURE IS USED TO SEARCH DATA ELEMENTS AND
' CHARACTER-STRINGS ACCORDING TO:
' 1.THE ENTRY POINTS PROVIDED BY THE ISEP PROCEDURE WHEN
'      IANA IS RUN FOR THE FIRST TIME,
' 2.THE IMPACT SEARCH CRITERIA PRODUCED
'      BY A PRECEDING EXECUTION OF IANA.
' IANA IS THEREFORE AN ITERATIVE PROCESS, WHICH RUNS
' UNTIL NO MORE IMPACT SEARCH CRITERIA ARE FOUND.
'
' ------------------------------------------------------
'
<job id=IANA>

<script language="VBScript">
Dim MyProc
MyProc = "IANA"
</script>

<script language="VBScript" src="INIT.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If

If Not FSO.FileExists(Rep_BASE & "\FO") Then
```

```
   Call Msg_Log (Array("1054"))
   Call Msg_Log (Array("1023"))
   Wscript.Quit (0)
End if

If Not FSO.FileExists(Rep_BASE & "\FQ") Then
   Call Msg_Log (Array("1053" , "FQ"))
   Call DisplayInfo (Msg)
   Wscript.Quit (0)
End if

Call Msg_Log (Array("1022" , "COPY"))
'-----------------------------------
Set MyFile = fso.GetFile(Rep_BASE & "\FQ")
MyFile.Copy (Rep_TMP & "\WFQ")
If Cobol = "Microfocus" then
   Set MyFile = fso.GetFile(Rep_BASE & "\FQ.idx")
   MyFile.Copy (Rep_TMP & "\WFQ.idx")
else 'acu
   Set MyFile = fso.GetFile(Rep_BASE & "\FQ.vix")
   MyFile.Copy (Rep_TMP & "\WFQ.vix")
end if

Call Msg_Log (Array("1022" , "PAN230"))
'-----------------------------------
WshEnv("PAC7MB") = Fic_Input
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
Call BvpEnv("PAN230","PAC7DD",Rep_USR & "\IANADD230.txt")
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7FH") = Rep_BASE & "\FH"
WshEnv("PAC7FR") = Rep_BASE & "\FR"
Call BvpEnv("PAN230","PAC7HF",Rep_TMP & "\WHF.tmp")
Call RunCmdLog ("BVPAN230")
If Return = 12 Then
Call Msg_Log (Array("1026" , "PAN230"))
End If
Call Err_Cod(Return , 0 , "PAN230")

Call Msg_Log (Array("1022" , "PAN220"))
'-----------------------------------
Call BvpEnv("PAN220","PAC7HF",Rep_TMP & "\WHF.tmp")
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
Call BvpEnv("PAN220","PAC7IL",Rep_USR & "\IANAIL220.txt")
Call RunCmdLog ("BVPAN220")
If Return = 12 Then
Call Msg_Log (Array("1026" , "PAN220"))
End If
Call Err_Cod(Return , 0 , "PAN220")


Call Msg_Log (Array("1022" , "PAN250"))
'-----------------------------------
```

```
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PAC7AY") = Rep_BASE & "\AY"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7FP") = Rep_BASE & "\FP"
WshEnv("PAC7FQ") = Rep_TMP & "\WFQ"
Call BvpEnv("PAN250","PAC7HF",Rep_TMP & "\WFH.tmp")
Call BvpEnv("PAN250","PAC7MF",Rep_TMP & "\WFO.tmp")
Call BvpEnv("PAN250","PAC7FH",Rep_TMP & "\WHF.tmp")
Call RunCmdLog ("BVPAN250")
If Return = 12 Then
Call Msg_Log (Array("1026" , "PAN250"))
End If
If Return = 4 Then
Call Msg_Log (Array("1055" , "PAN250"))
End If
Call Err_Cod(Return , 4 , "PAN250")

Call Msg_Log (Array("1022" , "PAN260"))
'-------------------------------------
Call BvpEnv("PAN260","PAC7MF",Rep_TMP & "\WFO.tmp")
WshEnv("PAC7OF") = Rep_BASE & "\FO"
WshEnv("PAC7FO") = Rep_BASE & "\FO-new"
Call RunCmdLog ("BVPAN260")
If Return = 12 Then
Call Msg_Log (Array("1026" , "PAN260"))
End If
Call Err_Cod(Return , 0 , "PAN260")

Call Msg_Log (Array("1022" , "PAN215"))
'-------------------------------------
Call BvpEnv("PAN215","PAC7FH",Rep_TMP & "\WFH.tmp")
WshEnv("PAC7HF") = Rep_BASE & "\FH-new"
WshEnv("PAC7FR") = Rep_BASE & "\FR-new"
Call BvpEnv("PAN215","PAC7FH",Rep_TMP & "\WFH.tmp")
Call RunCmdLog ("BVPAN215")
If Return = 12 Then
Call Msg_Log (Array("1026" , "PAN215"))
End If
Call Err_Cod(Return , 0 , "PAN215")

Call Msg_Log (Array("1022" , "COPY"))
'-----------------------------------
Set MyFile = fso.GetFile(Rep_TMP & "\WFQ")
MyFile.Copy (Rep_BASE & "\FQ-new")
If Cobol = "Microfocus" then
   Set MyFile = fso.GetFile(Rep_TMP & "\WFQ.idx")
   MyFile.Copy (Rep_BASE & "\FQ-new.idx")
else 'acu
   Set MyFile = fso.GetFile(Rep_TMP & "\WFQ.vix")
   MyFile.Copy (Rep_BASE & "\FQ-new.vix")
end if
```

```
Call Msg_Log (Array("1022" , "BACKUP"))
'-----------------------------------
Call Turnover(Rep_BASE & "\FH")
Call Turnover(Rep_BASE & "\FR")
Call Turnover(Rep_BASE & "\FO")
Call Turnover(Rep_BASE & "\FQ")


Call Msg_Log (Array("1024"))
'-------------------------------
Call DeleteFldr(Rep_TMP)

Call Msg_Log (Array("1023"))
'-----------------------------------
Wscript.Quit (Return)

</script>
</job>
```

IPFQ - FQ File Printout (Impact Analysis)

IPFQ - Introduction

The IPFQ procedure prints all the entry points and impact search criteria accepted or rejected during a comprehensive impact analysis.

All the criteria and entry points are stored in the FQ file.

IPFQ offers four types of printouts:

- List of accepted entry points
- List of rejected entry points
- List of accepted impact search criteria
- List of rejected impact search criteria.

The printout shows criteria and entry points sorted by alphabetical order within each category, and by definition Library of these criteria.

The printing order for the categories are:

- Character strings
- Element defined in the Dictionary,
- Element defined in Segment Descriptions,
- Element defined in Report Structures,
- Element defined in Screen- or Program-Working sections.

The IPFQ procedure can be used to select the entry points and impact search criteria of one or more categories.

In case of selection, only the selected criteria are printed.

Execution conditions

None, but the FQ file must exist.

Abnormal execution

Whatever the cause of the abnormal ending, the procedure can be restarted as it is, after the problem has been corrected.

Result

The procedure prints the entry points and the search criteria.

IPFQ - User Input

A '*' line with the user code and password.

One 'S' line per criteria selection (optional).

Position	Length	Value	Meaning
2	1	'S'	Line code
3	1		Type of criterion
		'E'	Element defined in the Dictionary
		'C'	Character string
		'X'	Group-type Element or Element not defined
		'*'	All types of criteria
4	1		Source code
		'3'	Line from Segment's -CE
		'6'	Line from Report's -CE
		'7'	-W line of a Screen or Program
		'*'	All sources
6	1		For the type of field
		'G'	For a Group field
		' '	For an elementary field
		'*'	For all types of fields

IPFQ - Description of Steps

Extraction of criteria: PAN240

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database Index file
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database Data file
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database Users
PAC7AR	Database dir. : AR	Input	Development Database Data file
PAC7AN	Database dir. : AN	Input	Development Database Index file
PAC7FQ	Database dir. : FQ	Input	Criteria impacted during analysis
PAC7MB	User input	Input	User input
PAC7FH	Tmp dir. : WFH	Output	Search criteria file
PAC7IX	User dir. : IPFQIX240	Report	Output report

Printing of impacted criteria: PAN220

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7HF	Tmp dir. : WFH	Input	Sorted entry points or criteria
PAC7IL	User dir. : IPFQIL220	Report	List of entry points and criteria

Return codes:

- 0 : OK.
- 12 : System error.

IPFQ - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'            - IMPACT ANALYSIS:  GROUP FIELDS ANALYSIS -
'
' ------------------------------------------------------
'
' THE IPFQ PROCEDURE PRINTS ALL THE ENTRY POINTS AND
```

```
' IMPACT SEARCH CRITERIA USED (ACCEPTED OR REJECTED)
' DURING A THOROUGH IMPACT ANALYSIS.
' ALL THE CRITERIA AND ENTRY POINTS ARE STORED IN THE FQ
' FILE.
' PROCEDURE, IF THEY ARE OF THE GROUP TYPE.
' ------------------------------------------------------
'
<job id=IPFQ>

<script language="VBScript">
Dim MyProc
MyProc = "IPFQ"
</script>

<script language="VBScript" src="INIT.vbs"/>


<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If

If Not FSO.FileExists( Rep_BASE & "\FQ") Then
   Call Msg_Log (Array("1053", "FQ"))
   Call Msg_Log (Array("1023"))
   Wscript.Quit (0)
End if

Call Msg_Log (Array("1022" , "PAN240"))
'-------------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
Call BvpEnv("PAN240","PAC7FH",Rep_TMP & "\WFH.tmp")
WshEnv("PAC7FQ") = Rep_BASE & "\FQ"
WshEnv("PAC7MB") = Fic_Input
Call BvpEnv("PAN240","PAC7IX",Rep_USR & "\IPFQIX240.txt")
Call RunCmdLog ("BVPAN240")
If Return = 12 Then
Call Msg_Log (Array("1026" , "PAN240"))
End If
Call Err_Cod(Return , 0 , "PAN240")

Call Msg_Log (Array("1022" , "PAN220"))
'-------------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
Call BvpEnv("PAN220","PAC7HF",Rep_TMP & "\WFH.tmp")
Call BvpEnv("PAN220","PAC7IL",Rep_USR & "\IPFQIL220.txt")
Call RunCmdLog ("BVPAN220")
If Return = 12 Then
Call Msg_Log (Array("1026" , "PAN220"))
End If
Call Err_Cod(Return , 0 , "PAN220")
```

```
Call Msg_Log (Array("1022" , "BACKUP"))
'-----------------------------------
Call Turnover(Rep_BASE & "\FQ")

Call Msg_Log (Array("1023"))
'-------------------------------
Call DeleteFldr(Rep_TMP)
Wscript.Quit (Return)

</script>
</job>
```

IPEP - Entry Points Printout

IPEP - Introduction

The IPEP procedure produces two types of printouts.

- List of entry points:

  This list is obtained after the ISEP procedure, since this procedure selects the entry points.
- List of impact search criteria:

  This list is obtained after the IANA procedure, since this procedure selects the impact search criteria.

In the printout, the criteria or entry points are sorted by alphabetical order (Elements and character strings altogether) for each definition library of these criteria.

The order of printing of the categories is:

- character string
- Element defined in Dictionary
- Element defined in Segment Description
- Element defined in Report Structure
- Element defined in the Screen or Program Working Section.

Execution conditions

None, but the FH file must exist.

Abnormal execution

Whatever the cause of the abend, the procedure can be restarted as it is, after the problem has been solved.

Printouts

Printout of entry points.

User input

No user input is required for the execution of the IPEP procedure.

IPEP - Description of Steps

Printing of entry points: PAN220

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7HF	Database dir. : FH	Input	Sorted entry points file
PAC7IL	User dir. : IPEPIL220	Report	List of entry points

.Return Codes :

- 0 : OK.
- 12 : System error

IPEP - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'     - IMPACT ANALYSIS:  PRINTING OF ENTRY POINTS -
'
' ------------------------------------------------------
'
' THE IPEP PROCEDURE PRODUCES TWO TYPES OF PRINTOUTS.
'   1. LIST OF ENTRY POINTS:
' THIS LIST IS OBTAINED AFTER THE ISEP PROCEDURE, SINCE
' THIS PROCEDURE SELECTS THE ENTRY POINTS.
'   2. LIST OF IMPACT SEARCH CRITERIA:
' THIS LIST IS OBTAINED AFTER THE IANA PROCEDURE, SINCE
' THIS PROCEDURE SELECTS THE IMPACT SEARCH CRITERIA.
' ------------------------------------------------------
'
<job id=IPEP>

<script language="VBScript">
Dim MyProc
MyProc = "IPEP"
</script>

<script language="VBScript" src="INIT.vbs"/>

<script language="VBScript">
```

```
If c_error = 1 then Wscript.Quit (1) End If

Call Msg_Log (Array("1022" , "PAN220"))
'------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7HF") = Rep_BASE & "\FH"
Call BvpEnv("PAN220","PAC7IL",Rep_USR & "\IPEPIL220.txt")
Call RunCmdLog ("BVPAN220")
If Return = 12 Then
Call Msg_Log (Array("1026" , "PAN220"))
End If
Call Err_Cod(Return , 0 , "PAN220")

Call Msg_Log (Array("1023"))
'----------------------------
Call DeleteFldr(Rep_TMP)
Wscript.Quit (Return)

</script>
</job>
```

IPIA - Printing of the Impact Analysis Results

IPIA - Introduction

The IPIA procedure is used to print Reports on the analysis results and to format these results in batch update transactions.

IPIA can output the following reports:

- Analysis results by entry point:

  Analysis follow-up of the subsequent iterations. Report requested by value '1' in Position 7 of the P-type user input line.
- List of impact search criteria by entry point:

  Valid when the IANA iteration is completed. Report requested by value '1' in Position 8 of the P-type user input line.
- Analysis results by Library:

  Results are formatted in batch update transactions (print or file output). Report requested by value '1' in Position 9 of the P-type user input line.

  Additional option (page and line skips) requested by value '2' in Position 9.

  File requested by value '1' in Position 12.
- Impacted-instances summary:

  List of all impacted instances with the number of impacted lines, for each type of line, not sorted by entry points. Report requested by value '1' in Position 10 of the P-type user input line.

- List of entry points by impacted search criterion for each impacted field: list of the entry points and impact search criteria which originated the impact, after each iteration. Report requested by value '1' in Position 14 of the P-type user input line.
- Statistics:

  Number of impacted lines sorted by library and by entity type, all lines considered. Report requested by value '1' in Position 11 of the P-type user input line.
- Character-string analysis:

  List of the uses of each character string searched by the ISOS procedure. Report requested by value '1' in Position 19 of the P-type user input line.
- Operator analysis:

  List of the uses of each operator searched by the ISOS procedure. Report requested by value '1' in Position 20 of the P-type user input line.
- List of the entities impacted by entry point:

  List of the entities impacted by Element-type entry points, all search criteria considered. Report requested by value '1' in Position 21 of the P-type user input line.
- Number of modified lines, dispatched by Description for each entity:

  This summary report allows for finer statistics by line types, compounded by Library. Report requested by value '1' in Position 22 of the P-type user input line.
- Constant analysis:

  List of uses of each constant searched by the ISOS procedure. Report requested by value '1' in Position 23 of the P-type user input line.

Execution conditions

None, but the FO file must exist and must not be empty.

Abnormal execution

Whatever the cause of the abend, the procedure can be restarted as it is after the problem has been solved.

Result

The procedure outputs a printout of the analysis results and of the list of transactions sorted by Library.

IPIA - User Input

A line identifying the context (* line) is required. It must be inserted at the beginning of the generated stream.

If you specified a lower library for the ISEP procedure, it must be repeated in this line.

The *-type line must be followed by one P-type, formatted as follows:

Position	Length	Value	Meaning
2	1	'P'	Line code
3	1		NOTHING TO ENTER,EXCEPT FOR DOS/VSE
		'I'	Default option for all hardware
		'N'	If CURRENT-DATE = MM/DD/YY
4	3	bbb	Library code (this selection is available with requests entered in Positions 9 and 10 only)
7	1	' '	No Result of impact analysis by entry point
		'1'	Result of impact analysis by entry point
8	1	' '	No List of impacted criteria by entry point
		'1'	List of impacted criteria by entry point
9	1	' '	No Printing of formatted results
		'1'	Printing of results formatted as batch update transactions, sorted per Library
		'2'	Same list with page and line skips
10	1	' '	No summary of impacted occurrences
		'1'	List of impacted instances
11	1	' '	No statistics, sorted per Library
		'1'	Statistics, sorted per Library
12	1	' '	Identical to values in Pos. 9 but output is a file instead of a print
13	1	' '	No inhibition of the lines indirectly impacted
		'1'	General option: Inhibition of the lines indirectly impacted (e.g. -CD)
14	1	' '	No list of entry points by impact
		'1'	List of entry points by impact search criterion
15	2	nn	Desired level number (IANA iteration)
17	2	pp	Number of lines printed per page

Position	Length	Value	Meaning
19	1	' '	No Result of character-string analysis
		'1'	Result of character-string analysis
20	1	' '	No Result of operator analysis
		'1'	Result of operator analysis
21	1	' '	No entities impacted by entry point
		'1'	Impacted entities by entry point
22	1	' '	No Number of lines per description
		'1'	Number of lines per description
23	1	' '	No result of constants analysis
		'1'	Result of constants analysis
24	1	' '	No Result of group fields analysis
24	1	'1'	Result of group fields analysis
25	10		Selection of generated transactions
		Blank	Selection of all entities
		other	Selection among the following entities (you can select several ones):
		'B'	Database blocks
		'E'	Elements
		'F'	Meta-Entities
		'O'	Screens, C/S Screens...
		'P'	Programs
		'R'	Reports
		'S'	Segments and Data-Structures
		'T'	Texts
		'V'	Documents
		'$'	User Entities
35	1	' '	No Recognition of ISOS transactions
		'1'	Recognition of ISOS transactions

IPIA - Description of Steps

Printing of impact results: PAN270

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages

Code	Physical name	Type	Label
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database Index file
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database Data file
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database Users
PAC7AR	Database dir. : AR	Input	Development Database Data file
PAC7FO	Database dir. : FO	Input	Impact results
PAC7MB	User input	Input	User input
PAC7BM	Tmp dir. : WMB	Output	Converted user input
PAC7GY	User dir. : IPIAGY	Output	PAF transactions for UPDP (length=310)
PAC7MV	Tmp dir. : WMV	Output	Batch transactions for printing (length=80)
PAC7IF	User dir. : IPIAIF270	Report	Analysis results

Return Codes :

- 0 : OK
- 12 : System error

Printing of generated transactions: PAN280

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7BM	Tmp dir. : WMB	Input	User input
PAC7MV	Tmp dir. : WMV	Input	Generated batch transactions
PAC7VM	User dir. : MVIPIA		Selected batch transactions (length=80)
PAC7IT	User dir. : IPIAIT280	Report	List of transactions per Library

Return Codes :

- 0 : OK
- 12 : System error

IPIA - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'        - IMPACT ANALYSIS : PRINTING OF RESULTS -
'
' ------------------------------------------------------
'
' THE IPIA PROCEDURE IS USED TO PRINT
' REPORTS ON THE ANALYSIS RESULTS
' AND TO FORMAT THESE RESULTS IN
' BATCH UPDATE TRANSACTIONS.
' ------------------------------------------------------
'
<job id=IPIA>

<script language="VBScript">
Dim MyProc
MyProc = "IPIA"
</script>

<script language="VBScript" src="INIT.vbs"/>


<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If

Call Msg_Log (Array("1022" , "PAN270"))
'-------------------------------------
WshEnv("PAC7MB") = Fic_Input
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7FO") = Rep_BASE & "\FO"
Call BvpEnv("PAN270","PAC7BM",Rep_TMP & "\WMB.tmp")
Call BvpEnv("PAN270","PAC7GY",Rep_USR & "\IPIAGY.txt")
Call BvpEnv("PAN270","PAC7MV",Rep_TMP & "\WMV.tmp")
Call BvpEnv("PAN270","PAC7IF",Rep_USR & "\IPIAIF270.txt")
Call RunCmdLog ("BVPAN270")
If Return = 12 Then
Call Msg_Log (Array("1026" , "PAN270"))
End If
Call Err_Cod(Return , 0 , "PAN270")

Call Msg_Log (Array("1022" , "PAN280"))
'-------------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
Call BvpEnv("PAN280","PAC7BM",Rep_TMP & "\WMB.tmp")
Call BvpEnv("PAN280","PAC7MV",Rep_TMP & "\WMV.tmp")
Call BvpEnv("PAN280","PAC7VM",Rep_USR & "\MVIPIA.txt")
Call BvpEnv("PAN280","PAC7IT",Rep_USR & "\IPIAIT280.txt")
```

```
Call RunCmdLog ("BVPAN280")
If Return = 12 Then
Call Msg_Log (Array("1026" , "PAN280"))
End If
Call Err_Cod(Return , 0 , "PAN280")

Call Msg_Log (Array("1024"))
'-----------------------------------
Call DeleteFldr(Rep_TMP)

Call Msg_Log (Array("1023"))
'----------------------------
Wscript.Quit (Return)

</script>
</job>
```

Chapter 8. Methodology Integrity Check

ADM - SSADM Pacdesign Methodology

SADM - Introduction

This procedure is available to the users who have purchased the SSADM Methodology Pacdesign module.

It checks the validity and consistency of occurrences uploaded (by the user) from the WorkStation to the VA Pacbase Repository.

NOTE: The SSADM Methodology and the procedure's functions exist in the English version only. For information on Pacdesign SSADM entities, consult the online help.

Execution conditions

None.

SADM - User Input

One '*' line for library access:

Position	Length	Value	Meaning
2	1	'*'	Line code
3	8	uuuuuuuu	User code
11	8	pppppppp	User password
19	3	bbb	Library code
22	4	nnnn	Session number (blank=current session)
26	1	'T'	Session status if Test session
37	25	.....	Reserved IMS: request identifier (cf. IMS BATCH PAF)

Print request lines:

Position	Length	Value	Meaning
2	1	'T'	Line code
3	1		Report code
		'V'	Validation of SSADM Entities

Position	Length	Value	Meaning
		'1'	Cross-boundaries Data flows within a DFD
		'2'	Operational Masters within a DSD
		'3'	All Entities with their attributes
4	6	eeeeee	Entity code (required for '1' or '2')

Printed output

This procedure prints the following, based on print requests:

- A 'Validation of SSADM entities' report,
- A 'List of cross-boundaries data flows within a DFD',
- A 'List of operational masters within a DSD',
- A 'List of all entities with their attributes'.

SADM - Description of Steps

SSADM-entity consistency check: PADM10

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database Index file
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database Data file
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database Users
PAC7AR	Database dir. : AR	Input	Development Database Data file
PAC7AN	Database dir. : AN	Input	Development Database Index file
PAC7MB	User input	Input	User input
SYSPAF	Tmp dir. : SYSPAF	Input/Output	Standard PAF indexed file
PAC7EJ	User dir. : SADMEJM10	Report	List of checked SSADM entities
PAC7ET	User dir. : SADMETM10	Report	PAF access report
PAC7DD	User dir. : SADMDDM10	Report	List of errors

SADM - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'           - PACDESIGN SSADM INTEGRITY CHECKING -
'
' ------------------------------------------------------
'
' THIS PROCEDURE IS SUPPLIED FOR USERS OF THE WORKSTATION
' AND THE SSADM PACDESIGN APPLICATION DESIGN METHODOLOGY.
'
' INPUT :
'  -  USER IDENTIFICATION LINE (REQUIRED)
'     COL  2 : "*"
'     COL  3 : USERIDXX
'     COL 11 : PASSWORD
'     COL 19 : (BBB)    LIBRARY CODE
'     COL 22 : (4 N)    SESSION NUMBER
'     COL 26 : (1 CAR.) SESSION VERSION
'     COL 37   (25 CAR.) RESERVED IMS
'  -  COMMAND LINE :
'  COL  2  : "T"        LINE CODE
'  COL  3  :  CODE FOR REPORT TO BE PRINTED
'                "V" : VALIDATION OF SSADM ENTITIES
'                "1" : CROSS-BOUNDARIES DATA FLOWS
'                      WITHIN A DFD
'                "2" : OPERATIONAL MASTERS WITHIN A DSD
'                "3" : ALL ENTITIES WITH THEIR ATTRIBUTES
'  COL  4  :  (6 CAR.) ENTITY CODE
'                      (REQUIRED FOR "1" OR "2")
'
' ------------------------------------------------------
'
<job id=SADM>

<script language="VBScript">
Dim MyProc
MyProc = "SADM"
</script>

<script language="VBScript" src="INIT.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If

Call Msg_Log (Array("1022" , "PADM10"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
```

```
Call BvpEnv("PADM10","PAC7DD",Rep_USR & "\SADMDDM10.txt")
Call BvpEnv("PADM10","PAC7EJ",Rep_USR & "\SADMEJM10.txt")
Call BvpEnv("PADM10","PAC7ET",Rep_USR & "\SADMETM10.txt")
WshEnv("PAC7MB") = Fic_Input
Call BvpEnv("PADM10","SYSPAF",Rep_TMP & "\SYSPAF.tmp")
Call RunCmdLog ("BVPADM10")

Call Err_Cod(Return , 0 , "PADM10")

Call Msg_Log (Array("1024"))
'----------------------------
Call DeleteFldr (Rep_TMP)


Call Msg_Log (Array("1023"))
'------------------------------
Wscript.Quit (Return)

</script>
</job>
```

YSMC - YSM Methodology / WorkStation

YSMC - Introduction

This procedure is available to the users who have purchased the YSM Methodology Pacdesign module.

- It checks the validity and the integrity of the entities uploaded from the WorkStation to the Host Specifications Dictionary by the user.
- It checks the consistency between a Data flow Diagram and its parent diagram.
- It establishes different hierarchical lists of certain entities of the Database.

Note

The YSM Methodology and the procedure's functions are available in the English version only.

For complete details, refer to the 'Pacdesign' manual.

Execution conditions

None.

YSMC - User Input

One '*'-line for library access (required):

Position	Length	Value	Meaning
2	1	'*'	Line code

Position	Length	Value	Meaning
3	8	uuuuuuuu	User code
11	8	pppppppp	User password
19	3	bbb	Code of the selected library
22	4	nnnn	Session number (space = current)
26	1	'T'	Session status if Test session
37	25	.....	Only for IMS: Request identifier (cf. PAF batch IMS)

Entity validation request line (optional):

Position	Length	Value	Meaning
2	1	'T'	Line code
3	1		Report code:
		'W'	'Validation of YSM entities'

PRC entity control request lines (optional):

Position	Length	Value	Meaning
2	1	'T'	Line code
3	1		Report code:
		'Y'	'Inter process consistency checking'
4	6	eeeeee	Entity code (PRC)

Print-request lines (optional):

Position	Length	Value	Meaning
2	1	'T'	Line code
3	1		Report code:
		'0'	'List of Relationships'
		'4'	'Process Decomposition list (CTX)'
		'5'	'Process Decomposition list (DFD)'
		'6'	'Datastore Decomposition list'
		'7'	'Event flow Decomposition list'
		'8'	'Group Data flow Decomposition list'
		'9'	'Multiple Data flow Decomposition list'

Position	Length	Value	Meaning
4	6	eeeeee	Entity code (REL/CTX/PRC/DST/EFL/DFL)

Printed report

This procedure prints:

- A 'Validation of YSM entities' report.
- An 'Inter-process consistency check' report.
- The reports:
  - 'List of relationships'.
  - 'Process decomposition list (CTX)'.
  - 'Process decomposition list (DFD)'.
  - 'Data store decomposition list'.
  - 'Event flow decomposition list'.
  - 'Group Data flow Decomposition list'.
  - 'Multiple Data flow Decomposition list'.

YSMC - Description of Steps

Validation of YSM entities: PYSMCC

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database Index file
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database Data file
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database Users
PAC7AR	Database dir. : AR	Input	Development Database Data file
PAC7AN	Database dir. : AN	Input	Development Database Index file
PAC7MB	User input	Input	User input
SYSPAF	Tmp dir. : SYSPAF	Input/Output	Standard PAF indexed file
PAC7EJ	User dir. : YSMCEJMCC	Report	Integrity checking lists
PAC7EI	User dir. : YSMCEIMCC	Report	Validation reports
PAC7DD	User dir. : YSMCDDMCC	Report	Error list

Validation of entities: PYSMC3

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database Index file
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database Data file
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database Users
PAC7AR	Database dir. : AR	Input	Development Database Data file
PAC7AN	Database dir. : AN	Input	Development Database Index file
PAC7MB	User input	Input	User input
SYSPAF	Tmp dir. : SYSPAF	Input/Output	Standard PAF indexed file
PAC7EJ	User dir. : YSMCEJMC3	Report	Integrity checking Lists

Validation of entities (2): PYSMC2

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database Index file
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database Data file
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database Users
PAC7AR	Database dir. : AR	Input	Development Database Data file
PAC7AN	Database dir. : AN	Input	Development Database Index file
PAC7MB	User input	Input	User input
SYSPAF	Tmp dir. : SYSPAF	Input/Output	Standard PAF indexed file
PAC7EJ	User dir. : YSMCEJMC2	Report	Integrity checking lists

YSMC - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
```

```
' ------------------------------------------------------
'          - PACDESIGN YSM INTEGRITY CHECKING -
'
' ------------------------------------------------------
'
' THIS PROCEDURE IS SUPPLIED FOR USERS OF THE WORKSTATION
' AND THE YSM PACDESIGN APPLICATION METHODOLOGY.
'
'
'     IT CHECKS THE VALIDITY AND THE INTEGRITY OF THE
' ENTITIES UPLOADED FROM THE WORKSTATION TO THE HOST
' SPECIFICATIONS DICTIONARY BY THE USER.
'     IT CHECKS THE CONSISTENCY BETWEEN A DATA FLOW
' DIAGRAM AND ITS PARENT DIAGRAM.(PRC)
'     IT ESTABLISHES DIFFERENT HIERARCHICAL LISTS OF
' CERTAIN ENTITIES OF THE DATABASE.
' ------------------------------------------------------
'
<job id=YSMC>

<script language="VBScript">
Dim MyProc
MyProc = "YSMC"
</script>

<script language="VBScript" src="INIT.vbs"/>

<script language="VBScript">

If c_error = 1 then Wscript.Quit (1) End If

Call Msg_Log (Array("1022" , "PYSMCC"))
'-------------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
Call BvpEnv("PYSMCC","PAC7DD",Rep_USR & "\YSMCDDMCC.txt")
Call BvpEnv("PYSMCC","PAC7EI",Rep_USR & "\YSMCEIMCC.txt")
Call BvpEnv("PYSMCC","PAC7EJ",Rep_USR & "\YSMCEJMCC.txt")
WshEnv("PAC7MB") = Fic_Input
Call BvpEnv("PYSMCC","SYSPAF",Rep_TMP & "\SYSPAF.tmp")
Call RunCmdLog ("BVPYSMCC")

Call Err_Cod(Return , 0 , "PYSMCC")

Call Msg_Log (Array("1022" , "PYSMC3"))
'-------------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
```

```
WshEnv("PACGGU") = Rep_ABASE & "\GU"
Call BvpEnv("PYSMC3","PAC7EJ",Rep_USR & "\YSMCEJMC3.txt")
WshEnv("PAC7MB") = Fic_Input
Call BvpEnv("PYSMC3","SYSPAF",Rep_TMP & "\SYSPAF.tmp")
Call RunCmdLog ("BVPYSMC3")
Call Err_Cod(Return , 0 , "PYSMC3")

Call Msg_Log (Array("1022" , "PYSMC2"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
Call BvpEnv("PYSMC2","PAC7EJ",Rep_USR & "\YSMCEJMC2.txt")
WshEnv("PAC7MB") = Fic_Input
Call BvpEnv("PYSMC2","SYSPAF",Rep_TMP & "\SYSPAF.tmp")
Call RunCmdLog ("BVPYSMC2")
Call Err_Cod(Return , 0 , "PYSMC2")

Call Msg_Log (Array("1024"))
'-----------------------------------
Call DeleteFldr (Rep_TMP)


Call Msg_Log (Array("1023"))
'-----------------------------------
Wscript.Quit (Return)


</script>
</job>
```

Part Number: DELNT003359A - 8591

Printed in USA