VisualAge Pacbase

The Developer's Procedures Windows 2000 or NT Server

Version 3.0

Note

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

Second Edition (November 2001)

This edition applies to the following licensed programs:
- VisualAge Pacbase Version 3.0

Comments on publications (including document reference number) should be sent electronically through the Support Center Web site at: http://www.ibm.com/software/ad/vapacbase/support.htm or to the following postal address:

IBM Paris Laboratory
1, place Jean–Baptiste Clément
93881 Noisy-le-Grand, France.

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

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.Cl´ment, 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.i

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 VisualAge Pacbase 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.

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.

Pos.	Len.	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
26	1	T	Test session
		H	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 extraction procedures generating update transactions which will modify a Library/session under DSMS control (you can also value them on the UPDT '*' line,
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:
		blanc	Replacement of the user code which locks the entity with 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
		blanc	The password is not transferred in the output file,
		1	The password is transferred, (Note : for EXTR, the '*' line is transferred in the output file only if you have entered a 'C' in Column 1)

Access Authorizations

The option requires a '*' line with user code and password as input of all procedures.

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

Abnormal Executions

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

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

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

In most cases, examining the status and type of operation allows the user 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
21	Sequence error
22	Duplicate key
23	No record found
24	Boundary violation (KSDS-RRDS)
30	System error
34	Boundary violation (sequential)
92	Logic error (For example, the opening of an already opened file
93	File still open on line
95	Invalid or Incomplete file

When this message is absent, and the type of ABEND generated directly reports a problem in the VisualAge Pacbase system programs, contact the VisualAge Pacbase support at IBM. KEEP ALL LISTINGS that may be necessary to analyze the problem.

Abends may my also occur during the execution of the procedure. For information, see the execution report of the procedure in the .LOG file and the VA Pac report of this same procedure.

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 !
```

Procedure : Definition and Execution

A procedure is a Windows Scripting script (.wsf) , including Visual Basic scripts (.vbs).

The commands file (.bvp) in input of the procedure is required. In this file, the * line contains the User code and the password.

Each procedure has arguments :

- the label of the Database concerned. (sub-directory of \DATA directory)
- The type of message display :
  - (1) : Display in a dialog box,
  - (2) : writing of the message in a xxxxMSG.LOGfile (default value),
  - (3) : inhibition of the messages.
- The name of the commands file (.bvp),which is by default under \DATA\[name_base]\INPUT. The input of the complete path allows to specify an other file.

For some procedures, there are no commands file but a data input file and output file. They are defined in standard under \TMP.

Temporary files, execution reportsand output files are found under directories created dynamically.

- Temporary files are located under: \DATA\[Base_name]\TMP\[user_code]\[proc_name]-[number]

- Output files, execution reports are located under:
  \DATA\[Base_name]\USERS\[user_code]\ [proc_name]-[number]

  This number is an application execution number, by default the process number of the procedure.

To execute a procedure, you may use different modes:

1. Execution using a command line :
   a. PROC.wsf [Base_name] [type_message] [Command_file]
   b. PRBVP.vbs [proc_name] [Base_name] [type_message] [Command_file]
2. Execution by double-clicking on :
   a. MBproc.bvp (in \DATA\(Base_name\INPUT)
   b. PRBVP.vbs (in \SYS\PROC) with input of the arguments : Proc_name, Base_name, Type_message.
3. Execution using the 'Start Menu' :
   - Access the shortcut : [[Base_name] Database Utilities] of the [VisualAge Pacbase Server\[Base_name]] group.
   - Enter the arguments : Proc_name, Type_message.

In all cases, the user code (and password) is(are) required.

Starting the On-Line Server

You must start the server to enable the connection of workstations and terminals to VisualAge Pacbase.

The shortcut [Start Base_name Database] located under the group of programs [VisualAge Pacbase Server] of the 'Start Menu', allows to start the server on the [Base_name] Database.

Connection of a 3270 Emulator

If you are working under Windows in character mode, you can access the VisualAge Pacbase server via a 3270 emulator installed on your PC.

The emulator must be configured accordingly, i.e., you must indicate the server port number and the address of the terminal on which the emulator is installed.

Chapter 2. Generation and Printing

GPRT : the Generation and Printing 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, 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 a unique program (BVPACB), which is used as a monitor calling the different programs that make up the procedure.

All programs that make up the procedure are thus considered to be sub-programs of this monitor, with which they communicate by means of a communication area and certain return codes.

Since user requests are often diverse, 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 families they manage. These families are identified by a one-position code as follows:

```
- A : Data Elements
- B : Database blocks
- C : COBOL programs
- D : Specifications Dictionary
- E : OLSD screens
- G : eBusiness Screens
- K : eBusiness Error messages
- L : Error messages (for OLSD)
- N : Personalized Documentation Manager
- P : Batch programs
- R : Production Environment Interface
- Q : Relational-SQL Database blocks
```

This code is referenced again in the names given to the programs, files and reports that are generated in this procedure. For programs, this is the sixth character of the code. Examples:

```
- BVPACA10 : General program.
- BVPACB30 : Database Block extractor.
```

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:

- Production Environment Interface,
- Database Blocks,
- COBOL programs (COB),
- On-line Screens (OLSD),
- Client Screens,
- Server Screens,
- Error Messages and Dialog Windowing,
- Personalized Documentation Manager,
- Batch programs,
- Specifications Dictionary.

Besides, a specific coding is used for files external names. It represents their use in the procedure:

- G : Generated code
- I : Reports
- J : Print requests
- K : Preparation for printing
- L : Error messages
- M : Transactions
- S : Skeletons
- W : Work

This code is found one character before last in the external name of the procedure files. Examples:

- PAC7GL : Generated error messages
- PAC7IN : Printing of Personalized Documentation

Files containing 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 file with an LG suffix, and is retrieved into the file with a GL suffix. This file is used to update the User Error Message file. It is used in input to the EMLD or EMUP procedures. Besides, these elements are printed in the IL-suffixed file.

The installed procedure does not include a name 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.

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 Guide.

GPRT : 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 listing the requests,
- All printing requested.

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

The GPRT procedure sends a general return code:

Code	Meaning
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 error messages
10	OK without generation
12	Input-Output error
16	Sort error

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		Ligne 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.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			(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.
			(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 PEI 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.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		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.
			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) This command is used to specify the job card and end-of-job delimiters: Flow control for Documents.
			Use the continuation line to define user parameters on the control cards.
		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. 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.
		LCE	(B) A list of defined Elements sequenced by Element code.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		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) Generate a COBOL description (COPY book) of the Data Structure.
			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).
		GED	(A) (D)
			C1 : Error messages generated for a Data Structure and for each Segment.
			C2 : Error messages generated through option 1 plus documentary help messages.
		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
		DCS	(B) (D: with input of the entity code) (3)
			NOTE: Enter the Data Structure code in the ENTITY CODE field, and the Segment code(s) on the continuation line(s).
			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.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			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.
		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.
		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.
		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.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		DCO	(A) Complete Screen Description including Dialog Complement and uses in other Screens. For Screens, information is also provided on relevant Segments, Macro-Structure Calls, Beginning Insertions Modifications, Work Areas and Structured Code.
			NOTE: To get the associated text, use print option '2'.
		DGC	(A) Complete Description of a C/S Screen.
		DGS	(A) Complete Description of a Business Component.
		DSO	(A) Description of the selected Screen.
		FGC	(C) (D) (4) (8) This command is used to specify the job card and end-of-job delimiters: Flow control for C/S Screens.
		FGE	(C) (D) (4) This command is used to specify the job card and end-of-job delimiters: Flow control for Pacbench/CS error messages.
			Use the continuation line to define user parameters on the control cards.
		FLE	(C) (D) (4) This command is used to specify the job card and end-of-job delimiters: Flow control for Dialog error messages.
			Use the continuation line to define user parameters on the control cards.
		FLO	(C) (D) (4) (8) This command is used to specify the job card and end-of-job delimiters: Flow control for Screens.
			Use the continuation line to define user parameters on the control cards.
		FSO	(C) (D) (4) (8) This command is used to specify the job card and end-of-job delimiters: Flow control for source Screen. Use the continuation line to define user parameters on the control cards.
		GCO	(A) (D) (5) Generate a COBOL Description of the Screen specified. Use the continuation line to define user parameters on the control cards.
		GGC	(A) (D) (5) Generate a C/S Screen (TUI Client Component).
		GGS	(A) (D) (5) Generation applicable to Business Component, Communication Monitor, Error Server, Folder.
		GEC	(A) (D) Pacbench C/S:
			C1 : Error messages defined for the Client or Server Dialog and for each component.
			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.
			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 Screen.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			C2 : Error messages generated through option 1 plus documentary help messages.
			C3 : Error messages for the Dialog only.
			C4 : Creation of the file required by Pacbase Web Connection. This command is applicable to the Dialog.
			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.
		GSO	Generate source code for the selected Screen.
		GVC	(A) (D) (5)
			Extract a Proxy object. Applicable to Folder View, Folder and Business Component.
		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 Transaction code.
		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: When requesting the Description of a single Report, enter the Report code prefix in the ENTITY CODE field and the last character of of the Report code on the continuation line.
			A complete Description of the Report(s). This include Report layouts. The information is sequenced by the Report code.
			Note: To get the associated text, use print option '2'
		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'.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		DSP	(A) Description of the selected Program produced by Reverse Engeneering.
		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) Generate a COBOL description of the Program specified Use the continuation line to define user parameters o the control cards.
		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. If no code is specified, all User Entities are listed. The information is sequenced by code.
		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 code.
		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) speci- fied.
			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.
			JCL INTRODUCTION
		JCL	This indicates that the COMMAND LABEL/SYSTEM RESPONSE field will contain JCL. The JCL command can only be entered in the 'C4' screen format option.
			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:
			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.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			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.
			METHOD ENTITIES PAF TABLES
		PCM	Description of PAF Tables for entities specific to a method. This command is necessarily followed by a Method code. VA PAC-GIP INTERFACE
		GIP	VisualAge Pacbase-GIP Interface generation.
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.
			'PCM' COMMAND: You enter in this field 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 4 options numbered from 1 to 4 (default option : 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.
17	2		Entity type
		O	Screen
		P	Program
		B	Database
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
			NOTE: This value may be overridden on the relevant entities' Definition screens. It may also be overridden at generation time.
20	1		CONTROL CARDS BEFORE MAP

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			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.
			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
			NOTE: This value may be overridden on the relevant entities' definitions screens. It may also be overridden at generation time.
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/

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 temporaty files is extremely variable.

Banners at the beginning and at the end of user documentation, which display the user code, facilitate the distribution of printouts back to their authors.

All programs, screens, Database Blocks, etc., which might be generated, are retrieved on GPRT.xx 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	Sorting error

GPRT : Execution Script

```
REM * -----------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -----------------------------------------------------
REM *    - GENERATION (IN INTERNAL READER) AND PRINTING -
REM *
REM * -----------------------------------------------------
REM *
REM *
REM * IN ADDITION TO THE GENERATED ENTITIES, THE FILE MUST
REM * CONTAIN THE JCL REQUIRED TO COMPILE THEM,
```

```
REM * USING THE BEGINNING/END OF JCL JOB STREAM OPTIONS AND
REM * THE BEFORE/AFTER PROGRAM OPTIONS.
REM *
REM * THE GENERATION AND PRINTING PROCEDURE, GPRT, HAS A
REM * TWO-FOLD PURPOSE:
REM *   . TO PRINT DOCUMENTATION USING  DATA CONTAINED IN THE
REM *       DATABASE, AND
REM *   . TO GENERATE PROGRAMS, SCREENS, DATABASE
REM *      DESCRIPTIONS DATA STRUCTURES,  AND ERROR MESSAGES.
REM *
REM * ------------------------------------------------------
REM *
<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_BVP & "_SYS\" _
& "\GENLANG")
End If


Call Msg_Log (Array("1022" , "PACB"))
'-------------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AJ") = Rep_BASE  & "\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("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7BM") = Rep_TMP & "\WWBM.tmp"
WshEnv("PAC7EB") = Rep_TMP & "\WWEB.tmp"
WshEnv("PAC7EE") = Rep_TMP & "\WWEE.tmp"
WshEnv("PAC7EG") = Rep_TMP & "\WWEG.tmp"
WshEnv("PAC7EI") = Rep_TMP & "\WWEI.tmp"
WshEnv("PAC7EN") = Rep_TMP & "\WWEN.tmp"
WshEnv("PAC7EP") = Rep_TMP & "\WWEP.tmp"
WshEnv("PAC7EQ") = Rep_TMP & "\WWEQ.tmp"
WshEnv("PAC7ER") = Rep_TMP & "\WWER.tmp"
WshEnv("PAC7EV") = Rep_TMP & "\WWEV.tmp"
WshEnv("PAC7EW") = Rep_TMP & "\WWEW.tmp"
WshEnv("PAC7GB")  = Rep_USR & "\GPRTGB.txt"
WshEnv("PAC7GD")  = Rep_USR & "\GPRTGD.txt"
WshEnv("PAC7GE")  = Rep_USR & "\GPRTGE.txt"
WshEnv("PAC7GF")  = Rep_USR & "\GPRTGF.txt"
WshEnv("PAC7GG")  = Rep_USR & "\GPRTGG.txt"
WshEnv("PAC7GI")  = Rep_USR & "\GPRTGI.txt"
WshEnv("PAC7GK")  = Rep_USR & "\ERRGK.txt"
WshEnv("PAC7GL")  = Rep_USR & "\ERRGL.txt"
WshEnv("PAC7GM")  = Rep_TMP & "\WXGM.tmp"
WshEnv("PAC7GN")  = Rep_TMP & "\WXGN.tmp"
WshEnv("PAC7GO")  = Rep_TMP & "\WWGO.tmp"
```

```
WshEnv("PAC7GP")  = Rep_USR & "\GPRTGP.txt"
WshEnv("PAC7GQ")  = Rep_USR & "\GPRTGQ.txt"
WshEnv("PAC7GR")  = Rep_USR & "\GPRTGR.txt"
WshEnv("PAC7GS")  = Rep_BASE  & "\GS"
WshEnv("PAC7GT")  = Rep_USR & "\PAWGT.txt"
WshEnv("PAC7GV")  = Rep_USR & "\GPRTGV.txt"
WshEnv("PAC7G6")  = Rep_USR & "\GPRTG6.txt"
WshEnv("PAC7DB")  = Rep_USR & "\GPRTDB.txt"
WshEnv("PAC7IA")  = Rep_USR & "\GPRTIA.txt"
WshEnv("PAC7ID")  = Rep_USR & "\GPRTID.txt"
WshEnv("PAC7IK")  = Rep_USR & "\GPRTIK.txt"
WshEnv("PAC7IL")  = Rep_USR & "\GPRTIL.txt"
WshEnv("PAC7IM")  = Rep_USR & "\GPRTIM.txt"
WshEnv("PAC7IN")  = Rep_USR & "\GPRTIN.txt"
WshEnv("PAC7JG")  = Rep_TMP & "\WWJG.tmp"
WshEnv("PAC7KB")  = Rep_TMP & "\WWKB.tmp"
WshEnv("PAC7KD")  = Rep_TMP & "\WWKD.tmp"
WshEnv("PAC7KE")  = Rep_TMP & "\WWKE.tmp"
WshEnv("PAC7KF")  = Rep_TMP & "\WWKF.tmp"
WshEnv("PAC7KG")  = Rep_TMP & "\WWKG.tmp"
WshEnv("PAC7KM")  = Rep_TMP & "\WWKM.tmp"
WshEnv("PAC7KN")  = Rep_TMP & "\WWKN.tmp"
WshEnv("PAC7KP")  = Rep_TMP & "\WWKP.tmp"
WshEnv("PAC7KQ")  = Rep_TMP & "\WWKQ.tmp"
WshEnv("PAC7KR")  = Rep_TMP & "\WWKR.tmp"
WshEnv("PAC7KS")  = Rep_TMP & "\WWKS.tmp"
WshEnv("PAC7KU")  = Rep_TMP & "\WWKU.tmp"
WshEnv("PAC7KV")  = Rep_TMP & "\WWKV.tmp"
'WshEnv("PAC7LG") =  Rep_USR & "\ERRLG.txt"
WshEnv("PAC7LG")  = "NUL"
WshEnv("PAC7LI")  = Rep_TMP & "\WWLI.tmp"
'WshEnv("PAC7LK")  = Rep_USR & "\ERRLK.txt"
WshEnv("PAC7LK")  = "NUL"
WshEnv("PAC7ME")  = Fic_Input
WshEnv("PAC7MG")  = Rep_TMP & "\WWMG.tmp"
WshEnv("PAC7MV")  = Rep_TMP & "\WWMV.tmp"
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("PAC7SO") = Rep_TMP & "\WSO.tmp"
WshEnv("PAC7SR") = Rep_SKEL & "\SR" & CodLang
WshEnv("PAC7SS") = Rep_SKEL & "\SS" & CodLang
WshEnv("PAC7SL") = Rep_SKEL & "\SL" & CodLang
WshEnv("PAC7WA")  = Rep_TMP & "\WWA.tmp"
WshEnv("PAC7W1")  = Rep_TMP & "\WW1.tmp"
WshEnv("PAC7W2")  = Rep_TMP & "\WW2.tmp"
WshEnv("PAC7W3")  = Rep_TMP & "\WW3.tmp"
WshEnv("PAC7W4")  = Rep_TMP & "\WW4.tmp"
WshEnv("PAC7W6")  = Rep_TMP & "\WW6.tmp"
WshEnv("PAC7W7")  = Rep_TMP & "\WW7.tmp"
WshEnv("PAC7W8")  = Rep_TMP & "\WW8.tmp"
WshEnv("PAC7W9")  = Rep_TMP & "\WW9.tmp"
WshEnv("SYSPAF")  = Rep_BASE & "\WPAF"
Return = WshShell.Run("BVPACB.exe" , 1, TRUE)

If Return < 10  then
Call Msg_Log (Array("1062"))
End if
If Return = 10  then
Call Msg_Log (Array("1063"))
End if
If Return > 10  then
Call Msg_Log (Array("1064"))
End if
Call Err_Cod(Return , 10 , "PACB")
```

```
If Return < 10 or  Return = 10  Then
'Information : Normal End
'-----------------------------------------
Return = 0
WshVolEnv("RC") = Return

Else
WshVolEnv("RC") = Return
Call Err_Cod(Return , 0 , "BVPACB")
End If


If  FSO.FileExists(Rep_USR & "\ERRGL.txt") Then
Set MyFile = fso.GetFile(Rep_USR & "\ERRGL.txt")
MyFile.Copy (Rep_USR & "\ERRLG.txt")
End If
If  FSO.FileExists(Rep_USR & "\ERRGK.txt") Then
Set MyFile = fso.GetFile(Rep_USR & "\ERRGK.txt")
MyFile.Copy (Rep_USR & "\ERRLK.txt")
End If


'PCM ...
'------------
 Dim Ig
 '9 files of Generation
 Ig = 9
 'Create GPRTXX  file
  Set LogGen = FSO.CreateTextFile(Rep_USR  _

  LogGen.Close
 'Write For Appending on GPRTXX.txt
 If  FSO.FileExists(Rep_USR & "\GPRTGP.txt") Then
 '-------------------------------------------------
  Set OutFil = FSO.GetFile(Rep_USR & "\GPRTXX.txt")
  Set W_OutFil = OutFil.OpenAsTextStream(ForAppending)
  Set R_InFil = FSO.OpenTextFile(Rep_USR & "\GPRTGP.txt" , ForRe
  Do While R_InFil.AtEndOfStream <> TRUE
     LineR = R_InFil.ReadLine
     W_OutFil.WriteLine (LineR)
  Loop
   R_InFil.Close
   W_OutFil.Close
 Else
  Ig = Ig - 1
 End If
 If  FSO.FileExists(Rep_USR & "\GPRTGQ.txt") Then
 '-------------------------------------------------
  Set OutFil = FSO.GetFile(Rep_USR & "\GPRTXX.txt")
  Set W_OutFil = OutFil.OpenAsTextStream(ForAppending)
  Set R_InFil = FSO.OpenTextFile(Rep_USR & "\GPRTGQ.txt" , ForRe
  Do While R_InFil.AtEndOfStream <> TRUE
     LineR = R_InFil.ReadLine
     W_OutFil.WriteLine (LineR)
  Loop
   R_InFil.Close
   W_OutFil.Close
 Else
  Ig = Ig - 1
 End If
 If  FSO.FileExists(Rep_USR & "\GPRTGR.txt") Then
 '-------------------------------------------------
  Set OutFil = FSO.GetFile(Rep_USR & "\GPRTXX.txt")
  Set W_OutFil = OutFil.OpenAsTextStream(ForAppending)
  Set R_InFil = FSO.OpenTextFile(Rep_USR & "\GPRTGR.txt" , ForRe
  Do While R_InFil.AtEndOfStream <> TRUE
     LineR = R_InFil.ReadLine
```

```
    W_OutFil.WriteLine (LineR)
 Loop
  R_InFil.Close
  W_OutFil.Close
Else
 Ig = Ig - 1
End If
If  FSO.FileExists(Rep_USR & "\GPRTGV.txt") Then
'-------------------------------------------------
 Set OutFil = FSO.GetFile(Rep_USR & "\GPRTXX.txt")
 Set W_OutFil = OutFil.OpenAsTextStream(ForAppending)
 Set R_InFil = FSO.OpenTextFile(Rep_USR & "\GPRTGV.txt" , ForRe
 Do While R_InFil.AtEndOfStream <> TRUE
    LineR = R_InFil.ReadLine
    W_OutFil.WriteLine (LineR)
 Loop
  R_InFil.Close
  W_OutFil.Close
Else
 Ig = Ig - 1
End If
If  FSO.FileExists(Rep_USR & "\GPRTGB.txt") Then
'-------------------------------------------------
 Set OutFil = FSO.GetFile(Rep_USR & "\GPRTXX.txt")
 Set W_OutFil = OutFil.OpenAsTextStream(ForAppending)
 Set R_InFil = FSO.OpenTextFile(Rep_USR & "\GPRTGB.txt" , ForRe
 Do While R_InFil.AtEndOfStream <> TRUE
    LineR = R_InFil.ReadLine
    W_OutFil.WriteLine (LineR)
 Loop
  R_InFil.Close
  W_OutFil.Close
Else
 Ig = Ig - 1
End If
If  FSO.FileExists(Rep_USR & "\GPRTGD.txt") Then
'-------------------------------------------------
 Set OutFil = FSO.GetFile(Rep_USR & "\GPRTXX.txt")
 Set W_OutFil = OutFil.OpenAsTextStream(ForAppending)
 Set R_InFil = FSO.OpenTextFile(Rep_USR & "\GPRTGD.txt" , ForRe
 Do While R_InFil.AtEndOfStream <> TRUE
    LineR = R_InFil.ReadLine
    W_OutFil.WriteLine (LineR)
 Loop
  R_InFil.Close
  W_OutFil.Close
Else
 Ig = Ig - 1
End If
If  FSO.FileExists(Rep_USR & "\GPRTGE.txt") Then
'-------------------------------------------------
 Set OutFil = FSO.GetFile(Rep_USR & "\GPRTXX.txt")
 Set W_OutFil = OutFil.OpenAsTextStream(ForAppending)
 Set R_InFil = FSO.OpenTextFile(Rep_USR & "\GPRTGE.txt" , ForRe
 Do While R_InFil.AtEndOfStream <> TRUE
    LineR = R_InFil.ReadLine
    W_OutFil.WriteLine (LineR)
 Loop
  R_InFil.Close
  W_OutFil.Close
Else
 Ig = Ig - 1
End If
If  FSO.FileExists(Rep_USR & "\GPRTGG.txt") Then
'-------------------------------------------------
 Set OutFil = FSO.GetFile(Rep_USR & "\GPRTXX.txt")
 Set W_OutFil = OutFil.OpenAsTextStream(ForAppending)
```

```
  Set R_InFil = FSO.OpenTextFile(Rep_USR & "\GPRTGG.txt" , ForRe
  Do While R_InFil.AtEndOfStream <> TRUE
     LineR = R_InFil.ReadLine
     W_OutFil.WriteLine (LineR)
  Loop
   R_InFil.Close
   W_OutFil.Close
 Else
  Ig = Ig - 1
 End If
 If  FSO.FileExists(Rep_USR & "\GPRTGF.txt") Then
 '-----------------------------------------------
  Set OutFil = FSO.GetFile(Rep_USR & "\GPRTXX.txt")
  Set W_OutFil = OutFil.OpenAsTextStream(ForAppending)
  Set R_InFil = FSO.OpenTextFile(Rep_USR & "\GPRTGF.txt" , ForRe
  Do While R_InFil.AtEndOfStream <> TRUE
     LineR = R_InFil.ReadLine
     W_OutFil.WriteLine (LineR)
  Loop
   R_InFil.Close
   W_OutFil.Close
 Else
  Ig = Ig - 1
 End If

  If Ig <> 0 Then
'Existing one or more file

 If Not FSO.FileExists(Rep_JOURNAL & "\QJ") Then
 Call Msg_Log (Array("1022" , "INQJ"))
'-------------------------------------
 WshEnv("PAC7QJ") = Rep_JOURNAL & "\QJ"
 Return = WshShell.Run("BVPCMINI.exe", 1, TRUE)
 Call Err_Cod(Return , 0 , "INQJ")
 End if

Call Msg_Log (Array("1022" , "PCM100"))
'-------------------------------------
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("PAC7GB")  = Rep_USR & "\GPRTXX.txt"
WshEnv("PAC7JG")  = Rep_TMP & "\WWJG.tmp"
WshEnv("PAC7ET") = Rep_USR & "\GPRTET100.txt"
WshEnv("PAC7BG") = Rep_USR & "\GPRTBG.txt"
WshEnv("PAC7QJ") = Rep_JOURNAL & "\QJ"
Return = WshShell.Run("BVPCM100.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PCM100")

End If


Call Msg_Log (Array("1024"))
'----------------------------
Call DeleteFldr(Rep_TMP)
Call DelFile (Rep_USR & "\GPRTXX.txt")

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

</script>
</job>
```

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 CONDITION

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

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

EMLD : User Input

One '*' line with user code and password.

EMLD : Description of Steps

INDEXED LOADING OF USER-DEFINED ERROR MESSAGES: PACL93

Code		Type	Label
PAC7MB		Input	Input Transactions
PAC7GL		Input	Sequential user-defined error messages
PACGGN		Input	Administration Database Index file
PACGGR		Entrée	Administration Database Data file
PACGGU		Input	Administration Database Users
PAC7AR		Input	Development Database Data file
PAC7AE		Input	Error messages
PAC7EM		Output	User-defined indexed error messages file
PAC7IY		Report	Output reports
PAC7DD		Report	Authorization option

Return code :

- 8 : no authorization on batch procedure.

EMLD : Execution Script

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

```
REM * -------------------------------------------------
REM *
<job id=EMLD>

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

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

<script language="VBScript">

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

If  FSO.FileExists(Rep_USR  & "\ERRGL.txt") Then

Call Msg_Log (Array("1022" , "PTUSGL"))
'-----------------------------------
WshEnv("PAC7LG") = Rep_USR  & "\ERRGL.txt"
WshEnv("PAC7GL") = Rep_USR  & "\ASCIIGL.txt"
Return = WshShell.Run("BVPTUSGL.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTUSGL")

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
WshEnv("PAC7GL") = Rep_USR & "\ASCIIGL.txt"
WshEnv("PAC7EM") = Rep_USR & "\ERRMSG.txt"
WshEnv("PAC7IY") = Rep_USR & "\EMLDIYL93.txt"
WshEnv("PAC7DD") = Rep_USR & "\EMLDDDL93.txt"
Return = WshShell.Run("BVPACL93.exe" , 1, TRUE)
If Return = 8 Then
Call Msg_Log (Array("1027"))
End If
Call Err_Cod(Return , 0 , "PACL93")

Else

Call Msg_Log (Array("1041" , Rep_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 output file of the GPRT procedure (GL-suffixed file) or from transactions for error message deletions at the entity level.

EXECUTION CONDITION

The User-Defined Error Message file must exist.

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

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

EMUP : User Input

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

Pos.	Len.	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:

Col.	Len.	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

UPDATE OF USER-DEFINED ERROR MESSAGES: PACL92

Code		Type	Label
PAC7GL		Input	Sequential user-defined error messages
PAC7AR		Input	Development Database Data file
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database index
PACGGR		Input	Administration Database Data file
PACGGU		Input	Administration Database users
PAC7MB		Input	Input transactions
PAC7EM		Output	User-defined error message indexed file
PAC7IU		Report	Transaction report
PAC7IX		Report	Error message report
PAC7DD		Report	Authorization option

Return code :

- 8 : no authorization on batch procedure.

EMUP : Execution Script

```
REM * -------------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -------------------------------------------------------
REM *             - USER'S ERROR MESSAGES UPDATING -
REM *
REM * -------------------------------------------------------
REM *
REM * THE EMUP PROCEDURE UPDATES THE USER-DEFINED ERROR
REM * MESSAGE FILE. THESE MESSAGES ARE OBTAINED FROM THE
REM * SEQUENTIAL OUTPUT FILE OF THE GPRT PROCEDURE (FILE WITH
REM * A GL SUFFIX) OR FROM TRANSACTIONS FOR ERROR
REM * MESSAGE DELETIONS AT THE ENTITY LEVEL.
REM *
REM * INPUT :
REM *  -  USER IDENTIFICATION LINE (REQUIRED)
REM *  -  COMMAND LINE :
REM *  COL 1  : 'D'    TRANSACTION CODE (DELETION)
REM *  COL 2  :  ENTITY TYPE; SAME AS IN CHOICE FIELD.
REM *            'O '   SCREEN
REM *            'D '   DATA STRUCTURE
REM *            'S '   SEGMENT
REM *  COL 4  : (6 CAR.) ENTITY CODE
REM * -------------------------------------------------------
REM *
<job id=EMUP>
<script language="VBScript">
MyProc = "EMUP"
</script>

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

<script language="VBScript">

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

If  FSO.FileExists(Rep_USR & "\ERRGL.txt") Then

Call Msg_Log (Array("1022" , "PTUSGL"))
'-----------------------------------
WshEnv("PAC7LG") = Rep_USR & "\ERRGL.txt"
WshEnv("PAC7GL") = Rep_USR  & "\ASCII.GL"
Return = WshShell.Run("BVPTUSGL.exe" , 1, TRUE)
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
WshEnv("PAC7GL") = Rep_USR & "\ASCIIGL.txt"
WshEnv("PAC7EM") = Rep_USR & "\ERRMSG.txt"
WshEnv("PAC7IU") = Rep_USR & "\EMUPIUL92.txt"
WshEnv("PAC7IX") = Rep_USR & "\EMUPIXL92.txt"
WshEnv("PAC7DD") = Rep_USR & "\EMUPDDL92.txt"
Return = WshShell.Run("BVPACL92.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PACL92")
If Return = 8 Then
Call Msg_Log (Array("1027"))
End If
Call Err_Cod(Return , 0 , "PACL92")
```

```
Else

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

End If

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

</script>
</job>
```

CODING OF GPRT OUTPUT FILES CREATED ON DISK

All output files generated by the GPRT procedure are created in the Temporary Files sub-directory (procedure 3rd parameter).

These files follow a special codification in order for the user to find his/her generated programs or reports easily.

GENERATED SOURCE AND PRINT FILES

These files are assigned the 'GPRT.' prefix:

```
      GENERATED SOURCE           PRINT FILES
      -------------------------  -----------------------
      GPRT.GB (Database Blocks)  GPRT.IA (Report)
      GPRT.GQ (SQL)
      GPRT.GD (Data)             GPRT.ID (Data)
      GPRT.GE (Screens - OSD)    GPRT.IH (PEI)
      GPRT.GP (Programs)         GPRT.IL (OSD Error Mes.)
      GPRT.GR (Reverse)          GPRT.IN (PDM-Volumes)
      GPRT.GG (Client screens)   GPRT.IK (OCS Error Mes.)
      GPRT.GV (Server screens)   GPRT.II (ICS Generat. Err)
      GPRT.GF (e-Business)
ERROR MESSAGE FILES
These files are assigned the "ERR." prefix:
      Input files : ERR.LG (OSD) and ERR.LK (OCS)
      Output files: ERR.GL (OSD) and ERR.GK (OCS)
```

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 two types of temporary files:

- Files internal to the GPRT procedure:

  These files are assigned the "WW" prefix and are deleted at the end of the procedure.
- Files which may be of interest to the VisualAge Pacbase user:

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\...).

These files are:

```
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 (CH: GP).

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 (FILES xxxxx.G6)

The files generated in RTF format on the VA Pac server may require to be converted into the ISO8859 character set and to the DOS format (for UNIX platforms) before being processed by the VA Pac WorkStation.

Conversion into ISO8859 character set:

cgitrans ″file .G6″ ″file for PDM/RTF conversion″ ″character set of the platform″ ″iso8859″

The character set of the platform can be the following one:

″hp″ on HP9000 platform

″pc850″ on AIX or OS2 platform using the PC850 set

Conversion into DOS file format:

cgiux2dos ″UNIX formatted file″ ″DOS formatted file″

These commands can be included in the PACAGP procedure.

SORT ON GENERATED SEQUENTIAL ERROR MESSAGES: PTUSGL

.Input file: -Sequential user error messages: PAC7LG (ERR.GL file in the temporary directory, output from GPRT)

.Output file: -Sorted sequential error messages: PAC7GL (ASCII.GL file in temporary directory)

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 program generation using the GPRT procedure, whose output is retrieved and used as input to PPAF, before compilation or storage 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 compilation.

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

Pos.	Len.	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
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

Pos.	Len.	Value	Meaning
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		Type	Label
PAC7AR		Input	Development Database Data file
PAC7AN		Input	Development Database Index file
PAC7AE		Input	Error messages
PAF80		Input	Generated programs
COB80		Output	Generated programs to be compiled
PAFREP		Report	Output report

PPAF : Execution Script

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

```
REM * -------------------------------------------------------
REM *
<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("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAF80") = Rep_USR & "\PPAFPAF.txt"
WshEnv("COB80") = Rep_USR & "\PPAFCOB.txt"
WshEnv("PAFREP") = Rep_TMP & "\WPAFREP.tmp"
Return = WshShell.Run("BVPAFP10.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PAFP10")

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


</script>
</job>
```

Chapter 3. Extractions

PACX : Introduction

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

See Chapter UPDP: Update from PAF extractions. This data is extracted in the form of transactions that can be used in input of the following procedures:

- UPDT
- UPDP
- CPSN (If the optional LCU 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

Pos.	Len.	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 explicits transaction codes
		' '	No formatting for UPDT
34	1	'1'	Formatting for UPDP (PAF)
		'2'	CPSN : formatting for UPDP with explicits transaction codes
		' '	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: user code = '*'-line user code
		'1'	No lock extraction
		'2'	Lock extraction: user code = original user code
		'N'	For RMEN only : no extraction of locked entities by an other user

Pos.	Len.	Value	Meaning
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) 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. 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

The 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 for purge (formatting for CPSN is not possible)
- RMEN: Extraction of entitites for upload/replacement/ recoding (formatting for CPSN is not possible). RMEN is subject to a separate purchase agreement.
- CPSN: comparison of sub-networks.

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.
- Formatting for CPSN: This procedure is part of the ' LCU Partitioned Database Manager' optional utility. Its use is therefore subject to a separate purchase agreement.
- Maximum number of input '*' cards : 1 for RMEN and EXPJ, 400 for EXSN, 1000 for EXTR, EXTA, EXUE et EXPU.

Printed result:

The PACX procedure produces:

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

Extraction of Entities

EXTR/EXTA : Introduction

These extractor types allow the selection of an entity as a whole or part of it.

If the request is of the '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 :

Pos.	Len.	Value	Meaning
2	1	'W'	Line code
3	1	'1'	Line number
4	2	'EX'	
6	1		Library selection code:
		'U'	Library alone
		'C'	Library and its upper-level libraries
7	33	'+' Choice	Library and its upper-level libraries with identification lines ('*' lines) generation Entity to be extracted, coded in the same way as the 'Choice' field in TP.
40	4		Extraction type:
		' '	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

Pos.	Len.	Value	Meaning
			'Ytt': Extension User Entity (tt = Meta-Entity type)

Second line (continuation line for selections and exceptions):

Pos.	Len.	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

(*): The EXTR procedure also works with choices that are specific to the Development Dtabase.

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 'multi-layered extractor' option ('ALL', 'EXPT' or 'ONLY' extraction type) is not available for EXTA. For this procedure, the value must be blank.
- If the extraction type is not specified, the extraction of a Data Structure extracts the Data Structure only. This field must therefore be completed if Segments are to be extracted also. Enter 'EXPTSEG' in this field.
- 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 'EXTPSEG' - 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.

Extraction of User Entities Contents

EXUE : Introduction

The EXUE procedure extracts the contents of User Entities according to their 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 'Personalization' manual.

EXUE : User Input - Printed Output - Result

USER INPUT

One command line per user entity:

Pos.	Len.	Value	Meaning
2	4	W1EX	Line code
6	1	$	Client UE Extraction identifier
		'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 112 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		Type	Label
PAC7AE		Input	Error messages
PAC7AN		Input	Development Database Index file
PAC7AR		Input	Development Database Data file
PAC7AY		Input	Development Database Extension Data
PACGGN		Input	Administration Database Index file
PACGGR		Input	Administration Database Data file
PACGGU		Input	Administration Database Users
PAC7PJ		Input	Archived transactions
PAC7MB		Input	User input

Code		Type	Label
PAC7MA		Input	CPSN Master file
PAC7ES		Input	CPSN Slave file
PAC7BM		Input/Output	User input
PAC7MM		Input/Output	EXPU Work file
PAC7MJ		Input/Output	EXPJ Work file
PAC7TE		Input/Output	RMEN Work file
PAC7RE		Input/Output	RMEN Work file
PAC7RM		Input/Output	RMEN Work file
PAC7WD		Input/Output	Extracted transactions
SYSEXT		Input/Output	Work file
PAC7MV		Output	Extracted transactions for UPDT
PAC7MR		Output	Extracted transactions for REOR (EXPU)
PAC7GY		Output	Extracted transactions for UPDP
PAC7TD		Output	Extracted transactions for CPSN
PAC7UE		Output	Extracted transactions for EXUE
PAC7IA		Report	General printout of the program stream
PAC7DD		Report	Errors on input transactions
PAC7ED		Report	Extractions report
PAC7EE		Report	Extractions report
PAC7EG		Report	Extractions report
PAC7EP		Report	Extractions report
PAC7EQ		Report	Extractions report
PAC7EU		Report	Extractions report
PAC7EZ		Report	Extractions report

Return codes :

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

PACX : Execution Script

```
REM * -------------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -------------------------------------------------------
REM *           - EXTRACTIONS FROM DATABASE -
REM *           - EXTRACTIONS COMPARATOR    -
REM * -------------------------------------------------------
REM *
REM * THE PACX PROCEDURE ALLOWS TO PERFORM VARIOUS TYPES
REM * OF DATA EXTRACTIONS FROM THE DEVELOPMENT DATABASE
REM * VIA PAF EXTRACTOR.
REM *
REM * POSSIBLE VALUES FOR THE EXTRACTOR CODE INCLUDE:
REM * - EXTR:  EXTRACTION OF ENTITIES
REM * - EXTA:  EXTRACTION OF ENTITIES (EXTRACTED TRANSACTIONS
```

```
REM *          ARE SORTED, ACCORDING TO THE INPUT
REM *          IDENTIFICATION LINES ORDER.
REM *          EACH REQUEST IS THUS PRECEDED BY A '*' LINE,
REM *          EXTRACTED TRANSACTIONS WILL BE SORTED IN THE
REM *          REQUEST ORDER).
REM * - EXUE:  EXTRACTION OF USER ENTITIES
REM * FOLLOWING VALUES ARE RESERVED FOR THE ADMINISTRATOR:
REM * - EXLI:EXTRACTION OF LIBRARIES OR LIBRARY SUB-NETWORKS
REM * - EXPJ:EXTRACTION OF JOURNAL (FORMATTING FOR CPSN IS
REM *        NOT POSSIBLE)
REM * - EXPU:EXTRACTION OF ENTITIES TO BE PURGED
REM *        (FORMATTING FOR CPSN IS NOT POSSIBLE)
REM * - RMEN:EXTRACTION OF ENTITIES FOR UPLOAD/REPLACEMENT/
REM *        RECODING (FORMATTING FOR CPSN IS NOT POSSIBLE).
REM *        RMEN IS SUBJECT TO A SEPARATE PURCHASE AGREEMENT
REM * - CPSN:COMPARISON OF SUB-NETWORKS.
REM *
REM * ------------------------------------------------------
REM *
<job id=PACX>

<script language="VBScript">
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
WshEnv("PAC7BM") = Rep_TMP & "\WBM.tmp"
WshEnv("PAC7WD") = Rep_TMP & "\WWD.tmp"
WshEnv("PAC7MM") = Rep_TMP & "\WMM.tmp"
WshEnv("PAC7MJ") = Rep_TMP & "\WMJ.tmp"
WshEnv("PAC7TE") = Rep_TMP & "\WTE.tmp"
WshEnv("PAC7RE") = Rep_TMP & "\WRE.tmp"
WshEnv("PAC7RM") = Rep_TMP & "\WRM.tmp"
WshEnv("PAC7MA") = "NUL"
WshEnv("PAC7ES") = "NUL"
WshEnv("PAC7UE") = Rep_USR & "\PACXUE.txt"
WshEnv("PAC7GY") = Rep_USR & "\PACXGY.txt"
WshEnv("PAC7TD") = Rep_USR & "\PACXTD.txt"
WshEnv("PAC7IA") = Rep_USR & "\PACXIA.txt"
WshEnv("PAC7DD") = Rep_USR & "\PACXDD.txt"
WshEnv("PAC7ED") = Rep_USR & "\PACXED.txt"
WshEnv("PAC7EE") = Rep_USR & "\PACXEE.txt"
WshEnv("PAC7EG") = Rep_USR & "\PACXEG.txt"
WshEnv("PAC7EP") = Rep_USR & "\PACXEP.txt"
WshEnv("PAC7EQ") = Rep_USR & "\PACXEQ.txt"
WshEnv("PAC7EU") = Rep_USR & "\PACXEU.txt"
WshEnv("PAC7EZ") = Rep_USR & "\PACXEZ.txt"
WshEnv("PAC7MV") = Rep_USR & "\PACXMV.txt"
WshEnv("PAC7MR") = Rep_USR & "\PACXMR.txt"
WshEnv("SYSEXT") = Rep_TMP & "\WSY.tmp"
Return = WshShell.Run("BVPACX.exe" , 1, TRUE)
```

```
Call Err_Cod(Return , 4 , "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 can be used separately or together.

PAF+ allows for the writing of the Extraction Master Path and for its execution when the PTEx is a User Extractor.

PDM+ allows for the writing and execution of the Master Outline (PTEd).

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

- When the PAF+/Extraction function is used alone, it allows for the generation of User Extractor programs with the possibility of formatting the data extracted.
- When the PDM+/Outline function is used alone, it allows for standardization skeletons (standard Print Options, Text instances always called, standardized calls).
- When both functions are used together, PAF+ extracts data from the Database. This data is processed by PDM+ and finally printed in a Document.

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 the Extraction Master Path

XPAF : Introduction

The Extraction Master Path validation procedure, XPAF, allows to perform specific extractions that the standard procedures are not able to perform. See the "Pacbase Access Facility (PAF)" 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 during a printing request by GPRT (choice: PCV).

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

PREREQUISITE

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

IMPLEMENTATION

Before the procedure can be executed, the user must define an instance of this meta-entity ($7E). 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

Pos.	Len.	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:

Pos.	Len.	Value	Meaning
2	2	'EX'	Line code
4	2		ME Type (7E by default)
6	6	eeeeee	User Entity code
			Warning: Specify library and session if the MEs whose instances are to be extracted 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. Check of the user entity's use in the sub-network. No update of GS if presence or use.

Examples

*user passwordLIB

EX7EEXT001_____UPDATE

*user passwordLIB

EX7EEXT002

XPAF : Description of Steps

ACCESS AND VALIDATION: PTEX30

Code		Type	Label
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database Index
PACGGR		Input	Administration Database Data file
PACGGU		Input	Administration Database Users
PAC7AN		Input	Development Database Index file
PAC7AR		Input	Development Database Data file
PAC7AY		Input	Development Database Extension Data
PAC7MB		Input	User Input
PAC7SP		Input	Variable skeleton file
PAC7GS		Input/Output	Extraction Paths
PAC7ED		Output	Report passed on to printing program
PAC7GP		Output	Temporary generated source
PAC7DD		Report	Report

EXTRACTION GENERATION: PTEX80

Code		Type	Label
PAC7AE		Input	Error Messages
PAC7SF		Input	Fixed skeleton file
PAC7GP		Input	Source file generated by PTEX30
PAC7ST		Output	Generated source to be translated

PREPROCESSOR: PAFP10

Code		Type	Label
PAC7AR		Input	Development Database Data file
PAC7AN		Input	Development Database Index file
PAC7AE		Input	Error messages
PAF80		Input	Generated programs
COB80		Output	Generated programs to be compiled
PAFREP		Report	Output report

PTEX PRINTING: PTEXD0

Code		Type	Label
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database Index file

Code		Type	Label
PACGGR		Input	Administration Database Data file
PACGGU		Input	Administration Database Users
PACGGY		Input	Administration Database Extension
PAC7ED		Input	PTEX30 Report
PAC7GS		Input/Output	Extraction Paths
PAC7RD		Report	Control report

XPAF : Execution Script

```
REM * -----------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -----------------------------------------------------
REM *                 - XPAF (PAF EXTENSION) -
REM *
REM * -----------------------------------------------------
REM *
REM * THE EXTRACTION MASTER PATH VALIDATION PROCEDURE,
REM * XPAF, ALLOWS FOR THE SIMULATION OF SPECIFIC EXTRACTIONS
REM * THAT THE STANDARD PROCEDURES ARE NOT ABLE TO PERFORM.
REM *
REM * INPUT :
REM *  -  USER IDENTIFICATION LINE (REQUIRED)
REM *     COL  2 : '*'
REM *     COL  3 : USERIDXX
REM *     COL 11 : PASSWORD
REM *     COL 19 : (BBB)    LIBRARY CODE
REM *     COL 22 : (4 N)    SESSION NUMBER
REM *     COL 26 : (1 CAR.) SESSION VERSION
REM *     COL 68 : ' '      STANDARD PRINT
REM *              '1'      UPPERCASE PRINT
REM *  -  COMMAND LINE :
REM *  COL  2  : 'EX'       LINE CODE
REM *  COL  4  :  (2 CAR.)  METAENTITY  TYPE (7E BY DEFAULT)
REM *  COL  6  :  (6 CAR.)  USER ENTITY CODE
REM *  COL 12  :  (BBB)     LIBRARY CODE     (IF THE U.E.O.
REM *  COL 15  :  (4 N)     SESSION NUMBER    ARE IN PARALLEL
REM *  COL 19  :  (1 CAR.)  SESSION VERSION   SUB-NETWORK)
REM *  COL 20  :  'UPDATE'  UPDATE OF GS
REM *              '      '  CHECK OF THE PRESENCE OF THE
REM *                        MASTER PATH IN GS.
REM *
REM * -----------------------------------------------------
REM *
<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"
WshEnv("PAC7ED") = Rep_TMP & "\WED.tmp"
WshEnv("PAC7DD") = Rep_USR & "\XPAFDDX30.txt"
WshEnv("PAC7GP") = Rep_TMP & "\WGP.tmp"
WshEnv("PAC7GS") = Rep_BASE & "\GS"
WshEnv("PAC7MB") = Fic_Input
WshEnv("PAC7SP") = Rep_SKEL & "\SP"
Return = WshShell.Run("BVPTEX30.exe" , 1, TRUE)
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"
WshEnv("PAC7GP") = Rep_TMP & "\WGP.tmp"
WshEnv("PAC7SF") = Rep_SKEL & "\SF"
WshEnv("PAC7ST") = Rep_TMP & "\WST.tmp"
Return = WshShell.Run("BVPTEX80.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTEX80")

Call Msg_Log (Array("1022" , "PAFP10"))
'-------------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PAF80") = Rep_TMP & "\WST.tmp"
WshEnv("COB80") = Rep_USR & "\COB80.txt"
WshEnv("PAFREP") = Rep_USR & "\PAFREP.txt"
Return = WshShell.Run("BVPAFP10.exe" , 1, TRUE)
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"
WshEnv("PAC7ED") = Rep_TMP & "\WED.tmp"
WshEnv("PAC7GS") = Rep_BASE & "\GS"
WshEnv("PAC7RD") = Rep_USR & "\XPAFRDXD0.txt"
Return = WshShell.Run("BVPTEXD0.exe" , 1, TRUE)
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 : Operations of the Extraction Master Path

EXECUTION OF A USER EXTRACTOR (E-TYPE PTEx)

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

User Input is described in the PAF Reference Manual, Chapter 'EXECUTION OF A USER EXTRACTOR / E-Type PTEx'.

For technical information on this execution, please refer to the Operations Manual, Part I, Chapter 'INSTALLATION', Subchapter 'COMPLEMENT: EXECUTION OF A PAF+ USER EXTRACTOR'.

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.

See the 'Personalized Documentation Manager' Manual for a complete documentation on the PDM+ Functionality.

NOTE:

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

Structuring Documentation - GDP+

XPDM : Validation of a Master Outline

XPDM : Introduction

A Master Outline is a P-type Document ('V' entity) designed to be called in another PDM Document. 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.

Pos.	Len.	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 :

Col.	Len.	Value	Meaning
2	2	'EP'	Line code
4	6	rrrrrr	Report 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		Type	Label
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database Index file
PACGGR		Input	Administration Database Data file
PACGGU		Input	Administration Database Users
PAC7AN		Input	Development Database Index file
PAC7AR		Input	Development Database Data file
PAC7AY		Input	Development Database Extension data
PAC7MB		Input	User Input
PAC7GS		Input Output	Extraction Paths
PAC7ED		Output	Report passed on to BVPTED80
PAC7SG		Output	GS-Update preparation

Code		Type	Label
PAC7DD		Report	Report

GS UPDATE AND PRINTING OF THE MASTER OUTLINE: PTED60

Code		Type	Label
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database Index file
PACGGR		Input	Administration Database Data file
PACGGU		Input	Administration Database Users
PACGGY		Input	Administration Database Extension
PAC7ED		Input	Print file
PAC7SG		Input	GS-Update preparation file
PAC7GS		Output	Extraction Paths
ETATGP		Report	Output report

XPDM : Execution Script

```
REM * -----------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -----------------------------------------------------
REM *                 - XPDM   (PDM EXTENSION) -
REM *
REM * -----------------------------------------------------
REM *
REM * A MASTER OUTLINE IS A P-TYPE VOLUME ('V' ENTITY)
REM * DESIGNED TO BE CALLED IN ANOTHER PDM VOLUME.
REM *
REM * INPUT :
REM *  -  USER IDENTIFICATION LINE (REQUIRED)
REM *     COL  2 : '*'
REM *     COL  3 : USERIDXX
REM *     COL 11 : PASSWORD
REM *     COL 19 : (BBB)    LIBRARY CODE
REM *     COL 22 : (4 N)    SESSION NUMBER
REM *     COL 26 : (1 CAR.) SESSION VERSION
REM *     COL 68 : ' '      STANDARD PRINT
REM *              '1'      UPPERCASE PRINT
REM *  -  COMMAND LINE :
REM *  COL  2  : 'EP'       LINE CODE
REM *  COL  4  :  (6 CAR.)  REPORT CODE
REM *  COL 10  :  'UPDATE'  UPDATE OF GS
REM *              '      '  CHECK OF THE VOLUME'S PRESENCE
REM *                        IN GS.
REM *
REM * -----------------------------------------------------
REM *
<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"
WshEnv("PAC7DD") = Rep_USR & "\XPDMDDD30.txt"
WshEnv("PAC7ED") = Rep_TMP & "\WED.tmp"
WshEnv("PAC7GS") = Rep_BASE & "\GS"
WshEnv("PAC7MB") = Fic_Input
WshEnv("PAC7SG") = Rep_TMP & "\WSG.tmp"
Return = WshShell.Run("BVPTED30.exe" , 1, TRUE)
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"
WshEnv("PAC7ED") = Rep_TMP & "\WED.tmp"
WshEnv("PAC7GS") = Rep_BASE & "\GS"
WshEnv("PAC7SG") = Rep_TMP & "\WSG.tmp"
WshEnv("ETATGP") = Rep_USR & "\XPDMGPD60.txt"
Return = WshShell.Run("BVPTED60.exe" , 1, TRUE)
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 contents of the PAC7GS file, where 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.

Pos.	Len.	Value	Meaning
2	1	'*'	Line code

Pos.	Len.	Value	Meaning
3	8	uuuuuuuu	User code
11	8	pppppppp	User password

PRGS : Description of Steps

!!!!Etat du fichier des plans types !

PRINTING OF THE MASTER PATH AND OUTLINE FILE:

Code		Type	Label
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database Index file
PACGGR		Input	Administration Database Data file
PACGGU		Input	Administration Database Users
PAC7GS		Input	Extraction Paths
PAC7MB		Input	User Input
PAC7DD		Report	Output Report
ETATGS		Report	Master Path and Outline file report

PRGS : Execution Script

```
REM * -----------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -----------------------------------------------------
REM *           -  PRINT OF MASTER PATH FILE -
REM *
REM * -----------------------------------------------------
REM *
REM * THE PRGS PROCEDURE PRINTS THE CONTENTS OF THE
REM * PAC7GS FILE, WHERE MASTER OUTLINES AND EXTRACTION
REM * MASTER PATHS ARE STORED.
REM *
REM * -----------------------------------------------------
REM *
<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("PAC7DD") = Rep_USR & "\PRGSDDP90.txt"
WshEnv("PAC7GS") = Rep_BASE & "\GS"
WshEnv("ETATGS") = Rep_USR & "\PRGSGSP90.txt"
WshEnv("PAC7MB") = Fic_Input
Return = WshShell.Run("BVPTEP90.exe" , 1, TRUE)
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 Extractions

UPDP : Introduction

The UPDP procedure performs an update of the Database from a sequential file reflecting 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

USER INPUT

The sequential file of input transactions is produced by a PAF extractor program. Its records mirror the PAF tables. For a detailed description of these tables, see the "Pacbase Access Facility Tables" manual.

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

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.

Pos.	Len.	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
		' '	current session
35	1	'T'	Session status: Test session
39	1	'A' or	Language code, useful if the
		'F'	transactions are not in the same language as the Database IN CASE OF A DSMS CONTROL OF THE DATABASE :
40	3	ppp	Product code

Pos.	Len.	Value	Meaning
43	6	nnnnnn	Product number

When the update is performed while the on line mode is active (on platforms that support this functionality), the input transaction flow must be preceded by a CHECKP table code line.

Pos.	Len.	Value	Meaning
2	10	'CHECKP'	Table code
12	4	nnnn	Number of transactions proces- sed between two pauses or checkpoints
16	4	'UPDT'	Update procedure
20	2	nn	LAN Platforms: Pause time, in seconds, bet- ween 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

Transactions formatting: PAF900

Code		Type	Label
PAC7AR		Input	Development Database Data file
PAC7AN		Input	Development Database Index file
PAC7AE		Input	Error messages
PACGGR		Input	Administration Database Data file
PACGGN		Input	Administration Database Index file
PACGGU		Input	Administration Database Users
PAC7GY		Input	Update transactions
PAC7MV		Output	Formatted transactions (must be able to contain all input transactions as well as elementary dletion transactions generated by multiple deletion transactions) (length=170)
PAC7ME		Output	Work file (length=372)
PAC7MW		Output	Work file (length=170)
PAC7MX		Output	Work file (length=743)
PAC7MY		Output	Work file (length=743)

Update of the Development Database : PACA15

Code		Type	Label
PAC7AR		Output	Development Database Data file

Code		Type	Label
PAC7AN		Output	Development Database Index file
PAC7AY		Output	Development Database Extension
PAC7AJ		Output	Development Database Journal
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database Index file
PACGGR		Input	Administration Database Data file
PACGGY		Input	Administration Database Extension
PACGGU		Input	Administration Database Users
PAC7DC		Input	Development Database elements DSMS file
PAC7ME		Input	Work file
PAC7MV		Input	Update transactions
PAC7RB		Output	UPDT erroneous transactions (length=80)
PAC7RY		Output	UPDP erroneous transactions (length=310)
PAC7IE		Report	Update report (length=132)
PAC7IF		Report	Summary 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 error
- 4 : fatal error

UPDP : Execution Script

```
REM * -----------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -----------------------------------------------------
REM *            - BATCH UPDATE FROM PAF TABLES -
REM *
REM * -----------------------------------------------------
REM *
REM * THE UPDP PROCEDURE PERFORMS AN UPDATE OF THE DATABASE
REM * FROM A SEQUENTIAL FILE REFLECTING PAF TABLES.
REM *
REM * THE SEQUENTIAL FILE OF INPUT TRANSACTIONS IS PRODUCED
REM * BY A PAF EXTRACTOR PROGRAM. ITS RECORDS MIRROR
REM * THE PAF TABLES.
REM * EACH SET OF TRANSACTIONS IMPACTING A LIBRARY OR SESSION
REM * MUST BE PRECEDED BY AN ASSIGN TABLE CODE LINE.
REM * WHEN THE UPDATE IS PERFORMED WHILE THE TP IS ACTIVE
REM * (ON PLATFORMS THAT SUPPORT THIS FUNCTIONALITY),
REM * THE INPUT TRANSACTION FLOW MUST BE PRECEDED BY A CHECKP
REM * TABLE CODE LINE.
REM * -----------------------------------------------------
REM *
<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", "PAF900"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PAC7GY") = Fic_Input
WshEnv("PAC7ME") = Rep_TMP & "\WME.tmp"
WshEnv("PAC7MW") = Rep_TMP & "\WMW.tmp"
WshEnv("PAC7MV") = Rep_TMP & "\WMV.tmp"
WshEnv("PAC7MX") = Rep_TMP & "\WMX.tmp"
WshEnv("PAC7MY") = Rep_TMP & "\WMY.tmp"
Return = WshShell.Run("BVPAF900.EXE" , 1, TRUE)
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("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7DC") = Rep_BASE & "\DC"
WshEnv("PAC7IE") = Rep_USR & "\UPDPIEA15.txt"
WshEnv("PAC7IF") = Rep_USR & "\UPDPIFA15.txt"
WshEnv("SEMLOCK") = Rep_BASE & "\LO"
WshEnv("SEMADMIN") = Rep_ABASE & "\LO"
WshEnv("PAC7ME") = Rep_TMP & "\WME.tmp"
WshEnv("PAC7MV") = Rep_TMP & "\WMV.tmp"
WshEnv("PAC7RB") = Rep_USR & "\UPDPRBA15.txt"
WshEnv("PAC7RY") = Rep_USR & "\UPDPRYA15.txt"
Return = WshShell.Run("BVPACA15.exe" , 1, TRUE)
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 libraries which make up the Database according to the different user authorizations.

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

EXECUTION CONDITIONS

The Database being 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.

IMPORTANT NOTES

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

- Back up, 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, Subchapter Database Restoration, in the ADMINISTRATOR'S GUIDE).

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 GUIDE).

ABNORMAL EXECUTION

Refer to Chapter OVERVIEW, Subchapter Abnormal Endings in the ADMINISTRATOR'S GUIDE.

There are two types of abnormal executions:

- Abnormal execution occurring 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 occurring during execution of the BVPACA15 program. The Database is left in an inconsistent state. If the problem appeared during input-output on a Database file, the printed error message and the file status will dictate the solution.

In either case, a restart can only take place after a restore using the Backup file including the transactions archived subsequent to this backup (REST procedure).

UPDT : User Input - Update Rules - Results

USER INPUT

A '*' line for user identification contains the user code, password and the corresponding library conversion.

The *-type line may contain conversion options: 'N' entered in column 67 inhibites any lowercase/uppercase conversion.

UPDATE RULES

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

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 on-line or in 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.

Note concerning deletion

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.

Warning

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

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 to create an incomplete Definition only for these entities' X-references (usually, User Entities) to be satisfied, a sort being impossible.

This code triggers the update of the Database.

The value 'P' allows to assign an identification line 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).

Multi-entity User Input

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

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

On a first line, you find the type and code of the entity concerned (with the line number if it is a description) and the line number for the comment.

- for the call of a P.I.A., the code of the entity is indicated on this first line and one single line is needed.
- for a comment line, there is a continuation line which bears the comment and at the end of the line, the type of line ('*' for Comments).

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.
			Note: 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	6		Code
12	3		Line number
			Numeric
15	60		Comment
75	1		Line Type
76	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

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		VC	Line code for a 'GC' screen
		VG	Line code for a 'GG' screen.
		VE	Line code for a 'GE' screen.
			Note: 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 that of the P.I.A. Description. The variable parts follow each other. There are no delimiters. The resolution includes the maximum length of each parameter defined.

Note

This line code comes always 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
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. Its contents depend upon the TYPE OF P.I.A. LINE.
			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 underscores (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
4	2		Entity type receiving the Comments
6	2		Meta-Entity Type
			It is an alphanumeric code entered at creation and which characterizes the Meta-Entity in all its types (two different Meta-Entities cannot have the same type) ; the type cannot be modified if User Entities have already been defined for this ME; this type, when used to define or describe a User Entity, is preceded by the '$' character (example: if the 'JOB' ME type is 'JO', the User Entities are referenced by '$JO.....').
8	30		Entity code (30 characters)
38	3		Line number
			PURE NUMERIC FIELD
			It is advisable to begin with line number '100' and then number in intervals of 20. This facilitates subsequent line insertions, as necessary.
41	6		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		Code ligne sur 1 caractère
		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.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		0	Program.
		U	User Manual.
		W1	Volume.
		Y1	User Entity.
		tt	tt User Entity Occurrences. Used for updating keywords of tt User Entity occurrences.
		Y5	User-Defined Relationship.
5	2		Meta-Entity Type
			It is an alphanumeric code entered at creation and which characterizes the Meta-Entity in all its types (two different Meta-Entities cannot have the same type) ; the type cannot be modified if User Entities have already been defined for this ME; this type, when used to define or describe a User Entity, 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.

To assign keywords, there is a continuation line which bears the keywords and at the end of the line, the

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.
80	1		Continuation line
		*	This value must be entered to indicate a continuation line.

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

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
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		Element Type
10	36		Name of Data Element
46	1		Type of format
		I	Internal format.
47	10		Data Element internal format
57	1		Element internal use
58	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		Element Type
10	36		Name of Data Element
46	1		Type of format
		E	Input format.
47	10		Conversational format
57	1		Element internal use
58	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	

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
3	6		Element code
9	1		Element Type
10	36		Name of Data Element
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
3	6		Element code
9	3		Line number
			Numeric
12	1		Line Type
13	1		Skip or action type
			Numeric
14	13		Data Element value
27	54		Dat Element value - Meaning

Model Objects

Definition (Line K1):

The access line used to define a model entity 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.).

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
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
		K3	
4	6		Object code
10	3		Line number
			Numeric
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 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

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			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
			Numeric
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
			Numeric
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 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
			Numeric
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

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
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 addressses 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
		E	
4	4		Segment code
8	1		TYPE OF SEGMENT DEFINITION LINE
		S	Sub-schema definition.
		Y	Sub-system definition.
12	10		Structure Code value
22	36		Label
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	NUMER.	Estimated number of instances

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			Numeric

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
			Numeric
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
			Optional
		O	Required Generation of a level 'E' (transaction refused) in standard error messages.
		P	Required_C Generation of a level 'C' (data element refused) in standard error messages.
		I	Forbidden
52	20		Modification
		F	Optional
		O	Required Generation of a level 'E' (transaction refused) in standard error messages.
		P	Required_C Generation of a level 'C' (data element refused) in standard error messages.
		I	Forbidden
72	1		Deletion
		F	Optional
		O	Mandatory Generation of a level 'E' (transaction refused) in standard error messages.
		P	Mandatory_C Generation of a level 'C' (data element refused) in standard error messages.
		I	Forbidden

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
73	1		Type 4
		F	Optional
		O	Required Generation of a level 'E' (transaction refused) in standard error messages.
		P	Required_C Generation of a level 'C' (data element refused) in standard error messages.
		I	Forbidden
74	1		Type 5
		F	Optional
		O	Required Generation of a level 'E' (transaction refused) in standard error messages.
		P	Required_C Generation of a level 'C' (data element refused) in standard error messages.
		I	Forbidden
75	1		Type 6
		F	Optional
		O	Required Generation of a level 'E' (transaction refused) in standard error messages.
		P	Required_C Generation of a level 'C' (data element refused) in standard error messages.
		I	Forbidden
76	1		Class (alpha/numeric)
77	1		Operators (and/or)
78	1		NEGATION (NOT)
		N	NEGATION ('NOT' is generated).
		blank	No negation.
79	1		Type: validation, update, values
80	13		Values / sub-function code
93	2		Update target / first part
95	2		Update target / second part
97	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

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		S	Sub-schema definition.
		Y	Sub-system definition.
9	1		Sub-schema / sub-system number
10	30		Sub-schema/sub-system clear 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		Name of report
36	36		Comments
72	1		Nature code
73	1		Type
74	3	NUMER.	Line length (maximum)
			Numeric
77	2	NUMER.	No. of digits left of the decimal
			Numeric
79	2	NUMER.	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
			Numeric
8	2	NUMER.	Constant part number
			Numeric

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
10	1		NUMBER OF PRINTED LITERALS PART
11	1		Line skip/page break
			Numeric
12	1		Character 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
6	2		Category (report)
8	3		Line number
			Numeric
14	1		Line Type
15	3		Length of the variable part
18	2	NUMER.	Structure number
			Numeric
20	2	NUMER.	Constant part number
			Numeric
22	2		Line skip/page break
			Numeric
24	1		Line skip type
25	2	NUMER.	Lines per page
			Numeric
27	2		Function code
29	2		Sub-function
31	2		Section priority
33	13		Comments
46	35		Conditions of report execution

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

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
2	1		Line code
		5	
3	3		Report Code
6	2		Category (report)
8	3		Line number
			Numeric
14	1		Line Type
15	3		Length of the variable part
18	2	NUMER.	Structure number
			Numeric
20	2	NUMER.	Constant part number
			Numeric
22	2		Line skip/page break
			Numeric
24	1		Line skip type
25	2	NUMER.	Lines per page
			Numeric
27	2		Function code
29	2		Sub-function
31	2		Section priority
33	13		Comments
46	35		Conditions of report execution

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	NUMER.	Structure number
			Numeric
8	3	NUMER.	Starting address (column number)
			Numeric
11	1		Data element line number
12	6		Element code
18	2		Continuation of D.S. Description

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		blank	First line of a Data Structure description. This line must contain all information defining the input-output characteristics, all technical characteristics and the description of the Data Structure.
			Two-letter code indicating a continuation line.
			The continuation lines are used to select the records of the different Data Structures in the Library and to request their description in a specified position.
20	14		Output Format
			(Default option: INTERNAL FORMAT)
			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 can also be used in a segment description.
			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 x(6)).
		C	With century (picture x(8)).
			Internal type formats:
		I	Without century (picture x(6)).
		S	With century (picture x(8)).
			Extended type formats (output) (with slashes):
		E	Without 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
40	6		Code of source field
46	3		Source field - last part

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
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 (col.80): the attributes, documentation call fields (PFkeys or characters), initialization character for entry fields (with NATURE = 'V'),
- With '*' in the continuation field (col.80): 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.

Note

on Screen Description lines ('I-type' access lines), enter the '?' character in column 31 to blank out the 'Type of Label' field.

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
			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

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
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
75	2		Screen type
77	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
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

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
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	2		Common area - data structure code
3	1		Organization
		D	IMS(DL/1), IDS I and IDS II Reserved for the description of Segments or records of the IMS (DL/1), IDS I or IDS II databases (depending on the 'Type of Cobol to Generate' in the Generation tab), in the generation of DBD, SYSGEN, schemas or application programs.
		Q	SQL/DS, DB2/2 or DB2/6000 Reserved for the description of SQL/DS, DB2/2 or DB2/6000 Databases (IBM), or ALLBASE/SQL Databases (HP3000), or DB2/2 or DB2/600 Databases (MICROFOCUS).
		2	DB2 or VAX/SQL Segment Generation-Description of a DB2 or VAX/SQL Segment. Only physical accesses are not generated. The structure of variable indicators corresponding to the columns of the DB2 or VAX/SQL table is always generated.
		O	ORACLE (< V6) Reserved for the description of an ORACLE (< V6) database structure.
		P	ORACLE (V6 and V7) Reserved for the description of an ORACLE (V6 and V7) database structure.
		R	RDMS Reserved for the description of an RDMS database structure.
		4	DB2/400 Reserved for the description of a DB2/400 database structure.
		N	NONSTOP SQL Reserved for the description of a NONSTOP SQL database structure.
		M	DATACOM DB Reserved for the description of a DATACOM DB database structure.
		9	INFORMIX, SYBASE, INGRES/SQL, or SQL SERVER Reserved for the description of an INFORMIX, SYBASE, INGRES/SQL, and SQL SERVER database structure.
		G	Table description This value generates the communication area with the Pactables access function.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
4	6		External name of error message file
10	4		First screen code of the dialogue
14	6		Database block code
20	4	NUMER.	Complementary common area length
			Numeric
24	47		Options

Description (Line I):

'I' is the line code used to describe a screen. It must be preceded by a line code H which specifies the dialogue Code.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		I	
3	3		Line number
			Numeric
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
26	1		Cursor default position/skip option
27	2	NUMER.	Horizontal repetitions
			Numeric
29	2	NUMER.	Vertical repetitions
			Numeric
31	1		Presence validation of data element
32	1		Validation conditions/set variables
33	1		Update option
34	4		Update target: segment code

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
38	6		Update target / last part
44	1		Working-Storage Prefix of Source
45	4		Source segment code
49	6		Code of source field
55	2		Level

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		I	
3	3		Line number
			Numeric
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
26	1		Cursor default position/skip option
27	2	NUMER.	Horizontal repetitions
			Numeric
29	2	NUMER.	Vertical repetitions
			Numeric
31	1		TYPE OF LITERAL
			Defines the contents of the next field, which is dis- played on the Call of Elements (-CE) with OPERATION = '2' (O: C2 or I2, etc.).
		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.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		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
			This group column contains the following elementary columns:
			CDSTPG, CRECPG.
9	2		Line number
			Numeric
11	1		ACCESS MODE
		S	Sequential (default option).
		R	Random - Direct (indexed sequential organization only).
			Note: With random access input files, the READ is not generated automatically.
		D	Dynamic (VSAM files only - ORGANIZATION = 'V')
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, CDELSR.
32	6		Element code
38	1		Control break indicator for display
39	1		Organization

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		D	IMS(DL/1), IDS I and IDS II Reserved for the description of Segments or records of the IMS (DL/1), IDS I or IDS II databases (depending on the 'Type of Cobol to Generate' in the Generation tab), in the generation of DBD, SYSGEN, schemas or application programs.
		Q	SQL/DS, DB2/2 or DB2/6000 Reserved for the description of SQL/DS, DB2/2 or DB2/6000 Databases (IBM), or ALLBASE/SQL Databases (HP3000), or DB2/2 or DB2/600 Databases (MICROFOCUS).
		2	DB2 or VAX/SQL Segment Generation-Description of a DB2 or VAX/SQL Segment. Only physical accesses are not generated. The structure of variable indicators corresponding to the columns of the DB2 or VAX/SQL table is always generated.
		O	ORACLE (< V6) Reserved for the description of an ORACLE (< V6) database structure.
		P	ORACLE (V6 and V7) Reserved for the description of an ORACLE (V6 and V7) database structure.
		R	RDMS Reserved for the description of an RDMS database structure.
		4	DB2/400 Reserved for the description of a DB2/400 database structure.
		N	NONSTOP SQL Reserved for the description of a NONSTOP SQL database structure.
		M	DATACOM DB Reserved for the description of a DATACOM DB database structure.
		9	INFORMIX, SYBASE, INGRES/SQL, or SQL SERVER Reserved for the description of an INFORMIX, SYBASE, INGRES/SQL, and SQL SERVER database structure.
		G	Table description This value generates the communication area with the Pactables access function.
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		Level

Call of Macro-Structures (Line M):

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

Since it contains no program code, this line code must always be preceded by line code '0' (Program Definition).

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

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		M	
3	2		Line number
			Numeric
5	1		Expansion option for Macro-Struct.
6	1		Delimiter of parameterized values
7	6		Macro-structure code
13	50		Parameter values
63	1		Continuation line

Program Beginning Insertions (Line D):

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

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		D	
3	2		Section to generate
5	2		Paragraph title
7	3		Line number
10	66		Instruction

Working Areas (Line 7):

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

Since it contains no program code, this line code must always be preceded by line code '0' (Program Definition).

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		7	
3	2		Line beginning
5	3		Line number
8	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.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		*	Comment. Data entered in the LEVEL AND SECTION and WORK AREA DESCRIPTION fields contain comments to be inserted into the generated Program (ANSI COBOL only).
		$	This value appears in column 7 of the generated COBOL and the other Elements of the WORKING line appear as it is.
		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.
			.LINE SEQUENCE.
			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 data-name entered in the WORK AREA DESCRIPTION must therefore have the following format:
			W-DDSS-EEEEEE where:
			W = a working-storage prefix,
			DDSS = a given DATA STRUCTURE and SEGMENT CODE,

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			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.
9	17		Level or section
26	48		Description
74	4		Table size (occurs clause)

Procedural Code (Line P):

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

Since it contains no program code, this form must always be preceded by line code '0' (Program Definition).

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		P	
3	2		Function code
5	2		Sub-function
7	3		Line number
10	1		Current processing Operators
11	32		Operand
43	2		Level
45	2		Condition type
47	32		Execution condition

Programs

Definition (Line O):

'0' (zero) is the line code used to define a program. It must always precede all description lines of a program (Batch 'D', 'M', '1' '7' 'P' and '9').

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		0	
3	6		Program code
9	6		Code for sequence of generation
15	30		Program name
45	1		Type of Cobol
46	1		ORDER OF INSERTION IN COBOL LIBRARY
		1	FIRST INPUT TO LIBRARY.
		2	AMENDMENT OF SOURCE LIBRARY.
			THIS OPTION MUST NOT BE USED IF OPTIONAL CONTROL CARDS ARE SPECIFIED.
		SPACE	NO INPUT TO SOURCE LIBRARY
			PL/1 : NOT USABLE, OPTIONAL CARDS BEFORE AND AFTER PROGRAM HAVE TO BE USED.
47	1		Cobol numbering and alignment option
		blank	Standard COBOL line Numbering, justification and identification of program in accordance with the standard COBOL line (default value).
		1	Suppression of numbering.
		2	Suppression of numbering and justification Suppression of numbering and justification of state- ments (columns 8 to 71 inclusive) in column 1.
		3	Suppression of program identification Standard numbering and justification, suppression of program identification.
		4	Suppression of numbering and program identification
		5	Suppression of numbering and of program identification Suppression of numbering and of program identifica- tion and justification of instructions (columns 8 to 71 inclusive) in column 1.
48	1		SQL generation option
49	1		Optional Command Lines Set/BEFORE
50	1		Optional Command Lines Set/AFTER
51	8		Cobol program id
59	1		Mode of programming
60	1		Type and structure of program
61	1		Type of presence validation
		blank	Present if not blank. It is the default value. The Data Element is present if its value is not blank.
		0	Present if not zero. The Data Element is present if its value is not zero.
		L	Present if not low-value. The Data Element is present if it does not contain low-values. This option is available for alphabetic and numeric Data Elements.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
62	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		Code ligne sur 1 caractère
		1	
3	2		Data structure code in the program
5	2		Data Structure code
7	6		External name
13	1		Organization
		D	IMS(DL/1), IDS I and IDS II Reserved for the description of Segments or records of the IMS (DL/1), IDS I or IDS II databases (depending on the 'Type of Cobol to Generate' in the Generation tab), in the generation of DBD, SYSGEN, schemas or application programs.
		Q	SQL/DS, DB2/2 or DB2/6000 Reserved for the description of SQL/DS, DB2/2 or DB2/6000 Databases (IBM), or ALLBASE/SQL Databases (HP3000), or DB2/2 or DB2/600 Databases (MICROFOCUS).
		2	DB2 or VAX/SQL Segment Generation-Description of a DB2 or VAX/SQL Segment. Only physical accesses are not generated. The structure of variable indicators corresponding to the columns of the DB2 or VAX/SQL table is always generated.
		O	ORACLE (< V6) Reserved for the description of an ORACLE (< V6) database structure.
		P	ORACLE (V6 and V7) Reserved for the description of an ORACLE (V6 and V7) database structure.
		R	RDMS Reserved for the description of an RDMS database structure.
		4	DB2/400 Reserved for the description of a DB2/400 database structure.
		N	NONSTOP SQL Reserved for the description of a NONSTOP SQL database structure.
		M	DATACOM DB Reserved for the description of a DATACOM DB database structure.
		9	INFORMIX, SYBASE, INGRES/SQL, or SQL SERVER Reserved for the description of an INFORMIX, SYBASE, INGRES/SQL, and SQL SERVER database structure.
		G	Table description This value generates the communication area with the Pactables access function.
14	1		Access mode
15	1		Recording mode

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		F	Fixed At generation time, the lengths of the different records are aligned with the length of the longest record.
		V	Variable
		U	Undefined
		S	Spanned This value is reserved for IBM MVS and DOS variants.
		C	with Database connection This value is reserved for Oracle V6/V7 and Sybase organizations. Automatic generation of CONNECT AT database, DECLARE database and access SQL AT database.
16	1		Opening mode
17	1		Unit type
18	5	NUMER.	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	NUMER.	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
54	4		Physical Unit Type
58	1		Unit Complement
			Unused
		R	Reader For IBM DOS.
		P	Punch For IBM DOS.
		S	EBCDIC set code For BULL DPS8 74.
		C	ASCII set code For BULL DPS8 74.
		O	OPTIONAL option not to be generated For BULL DPS7 and BULL DPS8 BCD.
		A	ALLOWING ALL file opening & REGARDLESS sequential read For DEC VAX VMS.
59	9		Sort key / seg select / report codes
68	1		Format type
69	1		Selected description
			No optional data elements are generated

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		V	All optional data elements are generated
		W	All optional data elements are generated, with ENPR
		E	Only ENPR and GRPR are generated
		U	Only ERUT is generated
		0	Subschema 0 generated
		1	Subschema 1 generated
		2	Subschema 2 generated
		3	Subschema 3 generated
		4	Subschema 4 generated
		5	Subschema 5 generated
		6	Subschema 6 generated
		7	Subschema 7 generated
		8	Subschema 8 generated
		9	Subschema 9 generated
70	1		Generated description type
71	1		Level
72	2		Line beginning
74	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 code, this line code must always be preceded by line code '0' (Program Definition).

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		M	
3	2		Line number
			Numeric
5	1		Expansion option for Macro-Struct.
6	1		Delimiter of parameterized values
7	6		Macro-structure code
13	50		Parameter values
63	1		Continuation line

Program Beginning Insertions (Line D):

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

Since it contains no program code, this line code must always be preceded by line code '0' (Program Definition).

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		D	
3	2		Section to generate
5	2		Paragraph title
7	3		Line number
10	66		Instruction

Working Areas (Line 7):

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

Since it contains no program code, this line code must always be preceded by line code '0' (Program Definition).

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		7	
3	2		Line beginning
5	3		Line number
8	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 (ANSI COBOL only).
		$	This value appears in column 7 of the generated COBOL and the other Elements of the WORKING line appear as it is.
		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).

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			.RECORD TYPE / USE WITHIN D.S. (I, E or S).
			.LEVEL NUMBER (COBOL) OF THE RECORD (1 to 5).
			.ORGANIZATION.
			.SUB-SCHEMA NUMBER.
			.LINE SEQUENCE.
			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 data-name entered in the WORK AREA DESCRIPTION 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.
9	17		Level or section
26	48		Description

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
74	4		Table size (occurs clause)

Procedural Code (Line P):

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

Since it contains no program code, this form must always be preceded by line code '0' (Program Definition).

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		P	
3	2		Function code
5	2		Sub-function
7	3		Line number
10	1		Current processing Operators
11	32		Operand
43	2		Level
45	2		Condition type
47	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 code must always be preceded by line code '0' (Program Definition).

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		FC	
4	2		Function code
6	2		Sub-function
8	3		Line number
11	1		Current processing Operators
12	67		Source line

Pur Cobol Source Lines (Line 9):

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

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	1		Line code
		9	
3	1		Continuation line
4	6		Cobol line number
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		Type of block
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 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

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
10	3		Line number
			Numeric
13	4		Child segment code
17	4		Parent segment code
21	6		Model Relation code
27	1		Identifier in Segment
28	9	NUMER.	Estimated number: child/parent links
			Numeric
37	36		Comment/relationship/key length
73	6		Path item (turboimage)
79	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		Type of block
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
2	2		Line code
		L3	
4	6		Database block code
10	3		Line number

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			Numeric
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	9	NUMER.	Estimated number: child/parent links
			Numeric
43	36		Comment/relationship/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		Type of block
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

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		L4	
4	6		Database block code
10	3		Line number
			Numeric
13	1		Structure code SQL batch transact
14	1		SQL RECORD TYPE
		P	Tablespace (except for INTEREL RDBC, INTEREL RFM, INGRES/SQL, DB2/400, VAX/SQL, NONSTOP SQL, INFORMIX, SYBASE and SQL SERVER)
		T	Table For ALLBASE/SQL, when a Primary Key or Foreign Key is defined in the Table (T line type) creation, the closing bracket must be entered on the line 690 of the -DRnnnG screen.
		V	View
		I	Index
		A	Alter Table: Column updating
		K	RDMS 1100, ALLBASE/SQL: Primary Key (Processed with the generation of the table that precedes it.)
			DB2, DATACOM/DB, INFORMIX-ESQL, 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, INFORMIX, SYBASE and SQL SERVER: Foreign key (Processed with the generation through an ALTER TABLE command.)
			ALLBASE/SQL: Foreign Key (Processed with the generation of the table that precedes it.)
		C	Package (ORACLE V7 only)
		E	Function (ORACLE V7 only)
		Q	Procedure (ORACLE V7, INGRES, SYBASE, SQL SERVER and INFORMIX)
		R	ORACLE V7, SYBASE and SQL SERVER: Trigger
			INGRES/SQL: Rule
15	18		Method external name
33	4		Segment code
37	1		Order
38	1		Key type
39	1		Type of generated transaction
40	6		Code of key data element no.1
46	1		Sort order 1
47	6		Code of key data element no.2
53	1		Sort order 2
54	6		Code of key data element no.3
60	1		Sort order 3
61	6		Code of key data element no.4

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
67	1		Sort order 4
68	6		Code of key data element no.5
74	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		Type of block
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 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
			Numeric
13	4		Child segment code
17	4		Parent segment code
21	6		Model Relation code
27	1		Identifier in Segment

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
28	9	NUMER.	Estimated number: child/parent links
			Numeric
37	36		Comment/relationship/key length
73	6		Path item (turboimage)
79	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
			Numeric
14	1		TYPE OF TEXT LINE
			SECTION TITLE
			A section must always contain a title. In batch, this title must be at the beginning of the section.
		L	Section title. It will NOT appear in an end-user documentation (User Manuals and Volumes).
		K	Same as type 'L' except that this title will appear in the end-user documentation (User Manuals and Volumes).
		-	Same as type 'K' but the title will be underlined with the '-' (dash) character when a Volume is printed.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		_	Same as type 'K' but the title will be underlined with the '_' (underscore) character when a Volume is printed.
		=	Same as type 'K' but the title will be underlined with the '=' character when a Volume is printed.
		+	Same as type 'K' but the title will be underlined with the '+' character when a Volume is printed.
			TEXT DESCRIPTION LINE
		blank	Default option.
			LINE/PAGE SKIP
			Taken into account when the text is printed in a User Manual or a Volume, or in Text simulation.
		1	New line.
		1-9	Skip of 1-8 lines before the given line is printed.
		*	PAGE skip before the given line is printed.
			TEXT ASSIGNMENT
		I	This code is used to assign a text 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.
			To facilitate data entry, key in an 'I' in this field and press ENTER. A dotted line will appear in the TEXT CONTENTS field to indicate where each entity type/ Instance code combination is to be entered.
		J	Explicit end of assignment.
			If there is no data in the TEXT CONTENTS field, the assignment of text to all entities is terminated.
			Using the same technique and format as with type 'I' above, the user may selectively end the text assignment by entity type/instance code, or by entity type.
			If no 'J' line is entered, the assignment goes to the end of the text.
		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.
			With the PACMODEL function:
		F D	Heading line allowing Activity Calculation. Detail line allowing Activity Calculation.
			NOTE: The L, I, J, Y, F and D Type lines are not printed in User Manuals or 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		Name of volume
46	1		Type of volume
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
			Numeric
17	1		Type of volume description line
18	1		Section level number
19	1		Line skip/page break
			Numeric
20	1		Character for title underlining
21	1		Print window
22	1		Alignment option

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
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		Code
10	36		Parameterized input aid clear name
46	1		Parameterized input aid type

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		Code
10	3		Line number
			Numeric
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		PACBASE TRANSACTION CODE
		SPACE	CREATION-MODIFICATION
		X	CREATION-MODIFICATION, WITH DATA CONTAINING AN AMPERSAND.
		C	CREATION
		M	MODIFICATION
		A	DELETION
		B	BEGINNING OF MULTIPLE DELETION
		R	END OF MULTIPLE DELETION
		S	STANDARD FUNCTION DELETION
2	2		Line code
		Y6	
4	6		Client Meta-Entity code
10	2		Description type
12	1		Description type
13	30		M.e. 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	

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

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

User Entities

Definition (Line Y3):

'Y3' is the line code used to define a UE.

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
		Y3	
4	2		Meta-entity calling code
6	6		User Entity short code
12	2		Range
14	1		Transaction number for User Entity
15	66		User Entity Definition Transaction

Description (Line Y4):

'Y4' is the line code used to describe the detail lines of a UE.

POS	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		Transaction code
2	2		Line code
		Y4	
4	2		Description number
6	6		User Entity descr. short identifier
12	2		Range
14	1		Transaction number for User Entity
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 S	Comments Synonyme(s)
19	55		Keyword description

UPDT : Description of Steps

Formatting transactions: PACA05

Code		Type	Label
PAC7AR		Input	Development Database Data file
PAC7AN		Input	Development Database Index file
PAC7AY		Input	Development Database extension data
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database Index file
PACGGR		Input	Administration Database Data file
PACGGU		Input	Administration Database Users
PAC7MB		Input	Update transactions
PAC7ME		Output	Work file (length=372)
PAC7MV		Output	Transactions formatted (length=170, may be able to contain all input transactions plus the elementary delete transactions generated by the multiple delete transactions)
PAC7MW		Report	Work file

Update of the development Database: PACA15

Code		Type	Label
PAC7AR		Output	Development Database Data file
PAC7AN		Output	Development Database index
PAC7AY		Output	Development Database extension
PAC7AJ		Output	Development Database journal
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database Index file
PACGGR		Input	Administration Database Data file
PACGGY		Input	Administration Database extension
PACGGU		Input	Administration Database users
PAC7DC		Input	DSMS file of Development Database Elements
PAC7ME		Input	Working file
PAC7MV		Input	Update transactions
PAC7RB		Output	UPDT erroneous transactions (length=80)
PAC7RY		Output	UPDP erroneous transactions (length=310)
PAC7IE		Report	Update report (length=132)
PAC7IF		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 without error
- 2 : warning
- 4 : fatal error

UPDT : Execution Script

```
REM * -------------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -------------------------------------------------------
REM *                 - BATCH UPDATE -
REM *
REM * -------------------------------------------------------
REM *
REM * REFER TO THE BATCH FORMS AND TO THE DESCRIPTION OF THE
REM * INPUT CORRESPONDING TO EACH ENTITY.
REM *
REM * INPUT :
REM *  -  USER IDENTIFICATION LINE (REQUIRED)
REM *     COL  2 : '*'
REM *     COL  3 : USERIDXX
REM *     COL 11 : PASSWORD
REM *     COL 28 : LANGUAGE CODE, USEFUL WHEN TRANSACTION ARE
REM *              NOT IN THE SAME LANGUAGE AS THE DATABASE.
REM *     COL 67 : 'N'  NOT 'UPPERCASE/LOWERCASE CONVERSION'
REM *  -  COMMAND LINE
REM *     THE LIST OF ALL AVAILABLE VALUES FOR THE ENTITY
REM *     TO BE UPDATED IS FOUND IN REFERENCE MANUAL.
REM *
REM * -------------------------------------------------------
REM *
<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", "PACA05"))
'-----------------------------------
WshEnv("PAC7MB") = Fic_Input
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("PAC7ME") = Rep_TMP & "\WME.tmp"
WshEnv("PAC7MV") = Rep_TMP & "\WMV.tmp"
WshEnv("PAC7MW") = Rep_TMP & "\WMW.tmp"
Return = WshShell.Run("BVPACA05.EXE" , 1, TRUE)
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("PAC7DC") = Rep_BASE & "\DC"
```

```
WshEnv("SEMLOCK") = Rep_BASE & "\LO"
WshEnv("SEMADMIN") = Rep_ABASE & "\LO"
WshEnv("PAC7IE") = Rep_USR & "\UPDTIEA15.txt"
WshEnv("PAC7IF") = Rep_USR & "\UPDTIFA15.txt"
WshEnv("PAC7ME") = Rep_TMP & "\WME.tmp"
WshEnv("PAC7MV") = Rep_TMP & "\WMV.tmp"
WshEnv("PAC7RB") = Rep_USR & "\UPDTRBA15.txt"
WshEnv("PAC7RY") = Rep_USR & "\UPDTRYA15.txt"
Return = WshShell.Run("BVPACA15.exe" , 1, TRUE)
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. Facility.

This interface extracts from the VisualAge Pacbase Database the descriptions of Tables necessary to 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 provide an interface file which is used as input to the GETT procedure of the Pactables Facility. For further details, refer to the Pactables Operations 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

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

Pos.	Len.	Value	Meaning
2	1	'*'	Line code
3	8	uuuuuuuu	User code
11	8	pppppppp	User password

Pos.	Len.	Value	Meaning
19	3	bbb	Library code
22	4	nnnn	Session number
26	1	t	Session status

One 'Z' line per generation or print request.

Pos.	Len.	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	DDMMCCYY	Date from which the table description can be modified. (Optional)
25	8	DDMMCCYY	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:
		' '	Internal format
		'E'	Input format
75	6	tttttt	Table number (if generating for a table other than that of the Segment's Definition file in the database).

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

NOTE: : Table keys cannot be modified: table generation requests applying to defined tables and involving 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		Type	Label
PAC7AR		Input	Development Database Data file
PAC7AN		Input	Development Database Index file
PAC7AY		Input	Development Database extension data
PAC7AE		Input	Error messages
PAC7TD		Input	Tables descriptions
PAC7MB		Input	Descriptions requests
PAC7MD		Output	Descriptions update transactions, version higher or equal to 2.0
PAC7ET		Report	Output report
PAC7DD		Input	Batch procedure authorization option

Return Codes : . 8 : No authorization on batch procedures.

FORMATTING OF DESCRIPTIONS < R 2.0: PACT45

Code		Type	Label
PAC7MD		Input	Description-update transactions higher or equal to 2.0
PAC7ND		Output	Description-update transactions lower or equal to 1.2

UPDATE OF TABLE-DESCRIPTION FILE: PACT50

(GETD procedure only)

Code		Type	Libellé
PACGGN		Input	Administration Database Index file
PACGGR		Input	Administration Database Data file
PACGGU		Input	Administration Database Users
PAC7TD		Input	Tables descriptions
PAC7MB		Input	Descriptions requests
PAC7MD		Input	Update transactions
PAC7ET		Report	Update report

GETD : Execution Script

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

```
REM * THE PACTABLES MANUAL.
REM *
REM * GETD IF THE DICTIONARY AND PACTABLES ARE RUNNING UNDER
REM * DIFFERENT ENVIRONMENTS.
REM *
REM * -----------------------------------------------------
REM *
<job id=GETD>

<script language="VBScript">
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("PAC7TD") = Rep_BASE & "\TD"
WshEnv("PAC7MB") = Fic_Input
WshEnv("PAC7MD") = Rep_USR & "\Mvgetd.txt"
WshEnv("PAC7ET") = Rep_USR & "\GETDETT40.txt"
WshEnv("PAC7DD") = Rep_USR & "\GETDDDT40.txt"
Return = WshShell.Run("BVPACT40.exe" , 1, TRUE)
If Return = 8 Then
Call Msg_Log (Array("1027"))
End If
Call Err_Cod(Return , 0 , "PACT40")

Call Msg_Log (Array("1022" , "PACT45"))
'-----------------------------------
WshEnv("PAC7ND") = Rep_USR & "\Ndgetd.txt"
Return = WshShell.Run("BVPACT45.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PACT45")

Call Msg_Log (Array("1022" , "PACT50"))
'--------------------------------
WshEnv("PAC7ET") = Rep_USR & "\GETDETT50.txt"
Return = WshShell.Run("BVPACT50.exe" , 1, TRUE)
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

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

<script language="VBScript">
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("PAC7TD") = Rep_BASE & "\TD"
WshEnv("PAC7MB") = Fic_Input
WshEnv("PAC7MD") = Rep_USR & "\Mvgeta.txt"
WshEnv("PAC7ET") = Rep_USR & "\GETAETT40.txt"
WshEnv("PAC7DD") = Rep_USR & "\GETADDT40.txt"
Return = WshShell.Run("BVPACT40.exe" , 1, TRUE)
If Return = 8 Then
Call Msg_Log (Array("1027"))
End If
Call Err_Cod(Return , 0 , "PACT40")

Call Msg_Log (Array("1022" , "PACT45"))
'-------------------------------------
WshEnv("PAC7ND") = Rep_USR & "\Ndgeta.txt"
Return = WshShell.Run("BVPACT45.exe" , 1, TRUE)
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

Pos.	Len.	Value	Meaning
2	1	'I'	Code carte
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 printing 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 DESCRIPTION FILE: PACTIN

Code		Type	Label
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database Index file
PACGGR		Input	Administration Database Data file
PACGGU		Input	Administration Database Users
PAC7AR		Input	Development Database Data file
PAC7MD		Input	Parameter line
PAC7TD		Output	Table description file
PAC7ED		Report	Initialization report

Code		Type	Label
PAC7DD		Report	Batch procedures authorization option

GETI : Execution Script

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

<script language="VBScript">
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("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7TD") = Rep_BASE & "\TD"
WshEnv("PAC7MB") = Fic_Input
WshEnv("PAC7ET") = Rep_USR & "\GETIEDTIN.txt"
Return = WshShell.Run("BVPACTIN.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PACTIN")

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.

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 VA Pac Database, by zooming on a given sub-network. 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.

  NOTE: Extraction of a session is also possible.

  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

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) instances of the entities.

For the FP file to be updated by INFP, you must re-state in the procedure's input all the lines previously introduced. You always start with an empty file, i.e. a file containing no particular selection.

RESULT

The procedure outputs a file which contains the entities selected for the analysis (FP).

INFP : User Input

Input is optional for the INFP procedure knowing that if no input is provided, all entities of all entity types will be searched for the impact analysis.

If all existing entities of a given entity type are specified (code = ******), particular entities specified for the same type will be refused.

If an entity type is specified in an input (whether an instance is specified or not for this type), and if you wish the analysis to tak into account the other types as well, you must specify those types in additional input lines.

Pos.	Len.	Value	Meaning
1	3		Entity type Possible values are:
		'B '	Database Blocks
		'F '	Meta-Entities
		'O '	Screens
		'P '	Programs
		'T '	Texts
		'V '	Documents
		'$nn'	User Entities of type code 'nn'
		'$**'	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		Type	Label
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database Index file
PACGGR		Input	Administration Database Data file
PACGGU		Input	Administration Database Users
PAC7AR		Input	Development Database Data file
PAC7MB		Input	User input
PAC7FP		Output	Entities in production
PAC7IP		Report	Check report

Return codes :

- 0 : OK
- 12 : System error

INFP : Execution Script

```
REM * -------------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -------------------------------------------------------
REM *     - IMPACT ANALYSIS:  INITIALIZATION OF 'FP' FILE -
REM *
REM * -------------------------------------------------------
REM *
REM * THE INFP PROCEDURE INITIALIZES THE FP FILE. IT ALLOWS
REM * TO SPECIFY THE ENTITIES WHICH ARE TO BE ANALYZED AND
REM * THUS TO NARROW THE SCOPE OF THE IMPACT ANALYSIS TO SOME
REM * (OR ALL) OCCURRENCES OF THE ENTITIES.
REM *
REM * -------------------------------------------------------
REM *
```

```
<job id=INFP>

<script language="VBScript">
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
WshEnv("PAC7IP") = Rep_USR & "\INFPIP.txt"
Return = WshShell.Run("BVPAN205.exe" , 1, TRUE)
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>
```

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').

Reports on entities using these operators and character-strings can be produced 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 VA Pac 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

Output of the ISOS procedure is:

- a 'FH' file (contains selected entry points),
- a 'FR' file (contains entry points to be purged),

  two files which are to be used in the IANA procedure,
- a 'FO' file (contains analysis results)

  to be used in the IANA or IPIA procedure.

ISOS : User Input

Only one '*'-line (required, placed at the beginning of the stream):

Pos.	Len.	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):

Pos.	Len.	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):

Pos.	Len.	Value	Meaning
2	1	'O'	Line code
3	3		Code of wanted operator (which must be recognized by the system)

One 'C'-line for the selection of character strings (optional):

Pos.	Len.	Value	Meaning
2	1	'C'	Line code
3	1		End-of-string separator (Required if the string contains at least one blank)
4	31		Code of searched string. (Must be ended by the separator if a sepa- rator 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 Report-specific Procedu- ral code 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):

Pos.	Len.	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 Report-specific Procedu- ral code 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		Type	Label
PAC7AE		Input	Error messages
PACGGN		Input	Administrator Database Index file
PACGGR		Input	Administrator Database Data file
PACGGU		Input	Administrator Database Users
PAC7AR		Input	Development Database Data file
PAC7AN		Input	Development Database Index file
PAC7FP		Input	Entities to analyze
PAC7MB		Input	User input
PAC7FH		Output	Selected entry points (length=160)
PAC7MF		Output	Impact analysis result (length=266)
PAC7IE		Report	Validation control

.Return Codes : . 0 : OK . 12 : System error

DELETION OF DUPLICATE ENTRY POINTS: PAN215

Code		Type	Label
PAC7FH		Input	Selected entry points
PAC7HF		Output	Sorted selected entry points
PAC7FR		Output	Reduced entry points to be purged

Return codes : . 0 : OK . 12 : System error

UPDATE OF IMPACT ANALYSIS RESULTS: PAN260

Code		Type	Label
PAC7MF		Input	Impact analysis result (for that iteration)

Code		Type	Label
PAC7OF		Input	Results from preceding analysis
PAC7FO		Input	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.OLD.

FR.NEW --(rotation)--> FR , the old FR file becomes FR.OLD.

FO.NEW --(rotation)--> FO , the old FO file becomes FO.OLD.

ISOS : Execution Script

```
REM * -----------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -----------------------------------------------------
REM * - IMPACT ANALYSIS: SELECTION OF STRINGS AND OPERATORS
REM *
REM * -----------------------------------------------------
REM *
REM * ISOS IS A COMPLEMENT TO THE ISEP PROCEDURE.
REM * FOR BETTER PERFORMANCE, IT IS ADVISABLE TO NARROW THE
REM * SCOPE OF THE SELECTION. THIS CAN BE DONE
REM * AT TWO DIFFERENT LEVELS, AND SHOULD ALWAYS
REM * BE DONE BEFORE RUNNING THE PROCEDURE.
REM *
REM * -----------------------------------------------------
REM *
<job id=ISOS>

<script language="VBScript">
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
WshEnv("PAC7FH") = Rep_TMP & "\WHF.tmp"
WshEnv("PAC7MF") = Rep_TMP & "\WMF.tmp"
WshEnv("PAC7IE") = Rep_USR & "\ISOSIE212.txt"
Return = WshShell.Run("BVPAN212.exe" , 1, TRUE)
If Return = 12 Then
```

```
Call Msg_Log (Array("1026" , "PAN212"))
End If
Call Err_Cod(Return , 0 , "PAN212")

Call Msg_Log (Array("1022" , "PAN215"))
'--------------------------------
WshEnv("PAC7HF") = Rep_BASE & "\FH.NEW"
WshEnv("PAC7FR") = Rep_BASE & "\FR.NEW"
WshEnv("PAC7FH") = Rep_TMP & "\WHF.tmp"
Return = WshShell.Run("BVPAN215.exe" , 1, TRUE)
If Return = 12 Then
Call Msg_Log (Array("1026" , "PAN215"))
End If
Call Err_Cod(Return , 0 , "PAN215")

Call Msg_Log (Array("1022" , "PAN260"))
'--------------------------------
WshEnv("PAC7OF") = Rep_BASE & "\FO"
WshEnv("PAC7FO") = Rep_BASE & "\FO.NEW"
Return = WshShell.Run("BVPAN260.exe" , 1, TRUE)
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>
```

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 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, placed at the beginning of the stream):

Pos.	Len.	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):

Pos.	Len.	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 inc- luded 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)

Pos.	Len.	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)

Pos.	Len.	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		Type	Label
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database Index file
PACGGR		Input	Administration Database Data file
PACGGU		Input	Administration Database Users
PAC7AR		Input	Development Database Data file
PAC7AN		Input	Development Database Index file
PAC7FP		Entrée	File of entities to be analyzed
PAC7MB		Input	User input
PAC7FH		Output	Selected entry points
PAC7IE		Report	Validation report

Return Codes :

. 0 : OK

. 12 : System error

REMOVAL OF DUPLICATE ENTRY POINTS: PAN215

Code		Type	Label
PAC7FH		Input	Selected entry points
PAC7HF		Input	Sorted selected entry points
PAC7FR		Input/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.OLD

FR.NEW --(rotation)--> FR , the old FR file becomes FR.OLD

ISEP : Execution Script

```
REM * -------------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -------------------------------------------------------
REM *         - IMPACT ANALYSIS : SELECTION OF ENTRY POINTS
REM *
REM * -------------------------------------------------------
REM *
REM * THE ISEP PROCEDURE IS DESIGNED TO SELECT THE ENTRY
REM * POINTS -- DATA ELEMENTS AND/OR CHARACTER STRINGS --
REM * WHICH WILL BE USED AS CRITERIA BY THE IMPACT
REM * ANALYSIS (IANA PROCEDURE).
REM *
REM * -------------------------------------------------------
REM *
<job id=ISEP>

<script language="VBScript">
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
```

```
WshEnv("PAC7FH") = Rep_TMP & "\WHF.tmp"
WshEnv("PAC7IE") = Rep_USR & "\ISEPIE210.txt"
Return = WshShell.Run("BVPAN210.exe" , 1, TRUE)
If Return = 12 Then
Call Msg_Log (Array("1026" , "PAN210"))
End If
Call Err_Cod(Return , 0 , "PAN210")

Call Msg_Log (Array("1022" , "PAN215"))
'---------------------------------
WshEnv("PAC7FH") = Rep_TMP & "\WHF.tmp"
WshEnv("PAC7HF") = Rep_BASE & "\FH.NEW"
WshEnv("PAC7FR") = Rep_BASE & "\FR.NEW"
Return = WshShell.Run("BVPAN215.exe" , 1, TRUE)
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>
```

IMFH : Merge of FH Files - Creation of FH and FR

IMFH : Introduction

The IMFH procedure allows you to merge two or more FH files so as to:

- Have only one FH file (selected entry points), after eliminating possible duplicates;
- Obtain a 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 a FH file.

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.

IMFH : Description of Steps

DELETION OF DUPLICATE ENTRY POINTS: PAN215

Code		Type	Label
PAC7FH		Input	Selected entry points to be merged
PAC7HF		Output	Sorted selected entry points
PAC7FR		Output	Reduced entry points to be purged

Return codes:

. 0 : OK . 12 : System error

IMFH : Execution Script

```
REM * -----------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -----------------------------------------------------
REM * - IMPACT ANALYSIS: MERGE FH FILES AND CREATION FR FILE
REM *
REM * -----------------------------------------------------
REM *
REM * THIS PROCEDURE SHOULD BE USED WHEN YOU WANT TO MERGE
REM * THE FH FILE PRODUCED BY THE ISEP PROCEDURE WITH THAT
REM * ISSUED BY THE ISOS PROCEDURE.
REM *
REM *
REM * -----------------------------------------------------
REM *
<job id=IMFH>

<script language="VBScript">
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"))
'--------------------------------
Set MyFile = fso.GetFile(Rep_BASE & "\FH.OLD")
MyFile.Copy (Rep_TMP  & "\WFH.tmp")
WshEnv("PAC7FH") = Rep_TMP  & "\WFH.tmp"
WshEnv("PAC7HF") = Rep_TMP  & "\WFHNEW.tmp"
WshEnv("PAC7FR") = Rep_BASE &  "\FR.NEW"
Return = WshShell.Run("BVPAN215.exe" , 1, TRUE)
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 DeleteFldr (Rep_USR)

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.

RESULT

The procedure outputs a reinitialized file of search criteria (FQ).

INFQ : Description of Steps

FQ file Reinitialization : PAN200

Code		Type	Label
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database Index file
PACGGR		Input	Administration Database Data file
PACGGU		Input	Administration Database Users
PAC7AR		Input	Development Database Data file
PAC7MB		Input	User input
PAC7FP		Output	Impacted criteria reinitialized sequential file
PAC7DD		Report	Error report

INFQ : Execution Script

```
REM * -----------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -----------------------------------------------------
REM *     - IMPACT ANALYSIS:  INITIALIZATION OF 'FQ' FILE -
REM *
REM * -----------------------------------------------------
REM *
REM * THIS ACTION SHOULD BE PERFORMED BEFORE A NEW IMPACT
REM * ANALYSIS EITHER BECAUSE THE ENTRY POINTS HAVE CHANGED
REM * OR BECAUSE THE ANALYSIS CONTEXT HAS CHANGED.
REM *
REM * -----------------------------------------------------
REM *
<job id=INFQ>
```

```
<script language="VBScript">
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"
WshEnv("PAC7DD") = Rep_USR  &  "\INFQDD200.txt"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7FP") = Rep_BASE & "\FQ"
Return = WshShell.Run("BVPAN200.exe" , 1, TRUE)
If Return = 12 Then
Call Msg_Log (Array("1026" , "PAN200"))
End If
Call Err_Cod(Return , 0 , "PAN200")

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

</script>
</job>
```

IGRA : Breaking down of Group Fields

IGRA : Introduction

The IGRA procedure breaks down group fields into Elementary Fields:

- Entry points detected by the ISEP procedure, if they are of the Group type.
- Impact search criteria obtained by running the IANA procedure, if they are of the Group type.

The IGRA procedure is optional and does not generate any impact search criterion.

Before running the IGRA procedure, you may purge:

- Entry points --after execution of the ISEP procedure.
- Impact search criteria --after execution of the IANA procedure which precedes.

In both cases, deletions are made in the FR file under 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.

USER INPUT

The IGRA procedure requires no specific user input for its execution.

RESULT

The procedure outputs a file which contains the analysis results (FO) to be used in the IPIA procedure.

IGRA : Description of Steps

RECOGNITION OF PURGED CRITERIA: PAN230

Code		Type	Label
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database Index file
PACGGR		Input	Administration Database Data file
PACGGU		Input	Administration Database Users
PAC7AR		Input	Development Database Data file
PAC7MB		Input	User input
PAC7FH		Input	Search criteria file
PAC7FR		Input	Reduced file of purged criteria

Code		Type	Label
PAC7HF		Output	Search criteria file (length=160)
PAC7DD		Report	Error file

PRINTING ENTRY POINTS: PAN220

Code		Type	Label
PAC7AE		Input	Error messages
PAC7HF		Input	Sorted criteria file
PAC7IL		Report	List of accepted/rejected criteria

GROUP FIELD BREAKING-DOWN: PAN255

Code		Type	Label
PACGGN		Input	Administration Database Index file
PACGGR		Input	Administration Database Data file
PACGGU		Input	Administration Database Users
PAC7AR		Input	Development Database Data file
PAC7AN		Input	Development Database Index file
PAC7FP		Input	Entities to analyze
PAC7FH		Input	Impacted criteria
PAC7MF		Output	Impact analysis results (length=266)

Return Codes :

- 0 : OK
- 12 : System error

UPDATE OF IMPACT ANALYSIS RESULTS: PAN260

Code		Type	Label
PAC7MF		Input	Impact analysis result (by level)
PAC7OF		Input	Results of previous analysis
PAC7FO		Output	Sorted results of the impact analysis

Return codes:

- 0 : OK.
- 12 : System error

IGRA : Execution Script

```
REM * -------------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -------------------------------------------------------
REM *            - IMPACT ANALYSIS:  PRINT OF 'FQ' FILE  -
REM *
REM * -------------------------------------------------------
REM *
```

```
REM * THE IGRA PROCEDURE BREAKS DOWN GROUP FIELDS
REM * INTO ELEMENTARY FIELDS:
REM * 1. ENTRY POINTS DETECTED BY THE ISEP PROCEDURE, IF
REM * THEY ARE OF THE GROUP TYPE.
REM * 2. IMPACT SEARCH CRITERIA OBTAINED BY RUNNING THE IANA
REM * PROCEDURE, IF THEY ARE OF THE GROUP TYPE.
REM *
REM * -------------------------------------------------------
REM *
<job id=IGRA>

<script language="VBScript">
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"
WshEnv("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"
WshEnv("PAC7HF") = Rep_TMP  &  "\WHF.tmp"
Return = WshShell.Run("BVPAN230.exe" , 1, TRUE)
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"
WshEnv("PAC7IL") = Rep_USR  & "\IGRAIL220.txt"
Return = WshShell.Run("BVPAN220.exe" , 1, TRUE)
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"
WshEnv("PAC7MF") = Rep_TMP  & "\WMF.tmp"
WshEnv("PAC7FH") = Rep_TMP  & "\WHF.tmp"
Return = WshShell.Run("BVPAN255.exe" , 1, TRUE)
If Return = 12 Then
Call Msg_Log (Array("1026" , "PAN255"))
End If
Call Err_Cod(Return , 0 , "PAN255")

Call Msg_Log (Array("1022" , "PAN260"))
'--------------------------------
```

```
WshEnv("PAC7MF") = Rep_TMP  & "\WMF.tmp"
WshEnv("PAC7OF") = Rep_BASE & "\FO"
WshEnv("PAC7FO") = Rep_BASE & "\FO.NEW"
Return = WshShell.Run("BVPAN260.exe" , 1, TRUE)
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 DeleteFldr (Rep_USR)

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 have the choice to inhibit:

- Entry points, after an 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 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 documented thereafter.

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 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

The IANA procedure does not require any specific user input.

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).

IANA : Description of Steps

RECOGNITION OF CRITERIA AFTER THE PURGE: PAN230

Code		Type	Label
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database Index file
PACGGR		Input	Administration Database Data file
PACGGU		Input	Administration Database Users
PAC7AR		Input	Development Database Data file
PAC7MB		Input	User input
PAC7FH		Input	Search criteria file
PAC7FR		Input	Search criteria after purge (reduced file)
PAC7HF		Ouput	Search criteria file (length=160)
PAC7DD		Report	Error report

PRINTING OF ENTRY POINTS: PAN220

Code		Type	Label
PAC7AE		Input	Error messages
PAC7HF		Input	Sorted criteria
PAC7IL		Output	List of accepted / rejected criteria

IMPACT ANALYSIS: PAN250

Code		Type	Label
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database Index file
PACGGR		Input	Administration Database Data file
PACGGU		Input	Administration Database Users
PAC7AR		Input	Development Database Data file
PAC7AN		Input	Development Database Index file
PAC7AY		Input	Development Database extension data
PAC7FP		Input	File of entities to be analyzed
PAC7FH		Input	Impacted criteria
PAC7FQ		Input/Output	Impacted criteria already processed
PAC7HF		Output	New impacted criteria (length = 160)
PAC7MF		Output	Impact analysis results (length = 266)

Return codes:

. 0 : OK

. 12 : System error

UPDATE OF IMPACT ANALYSIS RESULTS: PAN260

Code		Type	Label
PAC7MF		Input	Impact analysis results (level)
PAC7OF		Input	Results of previous analysis
PAC7FO		Output	Sorted results of impact analysis

Return codes:

. 0 : OK

. 12 : System error

REMOVAL OF DUPLICATE ENTRY POINTS: PAN215

Code		Type	Label
PAC7FH		Input	Selected entry points
PAC7HF		Output	Sorted selected entry points
PAC7FR		Output	Reduced entry points to be purged

Return codes:

. 0 : OK.

. 12 : System error

IANA : Execution Script

```
REM * -----------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -----------------------------------------------------
REM *                  - IMPACT ANALYSIS -
REM *
REM * -----------------------------------------------------
REM *
REM * THE IANA PROCEDURE IS USED TO SEARCH DATA ELEMENTS AND
REM * CHARACTER-STRINGS ACCORDING TO:
REM * 1.THE ENTRY POINTS PROVIDED BY THE ISEP PROCEDURE WHEN
REM *      IANA IS RUN FOR THE FIRST TIME,
REM * 2.THE IMPACT SEARCH CRITERIA PRODUCED
REM *      BY A PRECEDING EXECUTION OF IANA.
REM * IANA IS THEREFORE AN ITERATIVE PROCESS, WHICH RUNS
REM * UNTIL NO MORE IMPACT SEARCH CRITERIA ARE FOUND.
REM *
REM * -----------------------------------------------------
REM *
<job id=IANA>

<script language="VBScript">
MyProc = "IANA"
</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"
WshEnv("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"
WshEnv("PAC7HF") = Rep_TMP  &  "\WHF.tmp"
Return = WshShell.Run("BVPAN230.exe" , 1, TRUE)
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"
WshEnv("PAC7HF") = Rep_BASE &  "\FH"
WshEnv("PAC7IL") = Rep_USR  &  "\IANAIL220.txt"
Return = WshShell.Run("BVPAN220.exe" , 1, TRUE)
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("PAC7FQ") = Rep_BASE & "\FQ"
WshEnv("PAC7FP") = Rep_BASE & "\FP"
Set MyFile = fso.GetFile(Rep_BASE & "\FQ")
MyFile.Copy (Rep_BASE & "\FQ.NEW")
Set MyFile = fso.GetFile(Rep_BASE & "\FQ.idx")
MyFile.Copy (Rep_BASE & "\FQ.NEW.idx")
WshEnv("PAC7FQ") = Rep_BASE & "\FQ.NEW"
WshEnv("PAC7HF") = Rep_TMP  &  "\WFH.tmp"
WshEnv("PAC7MF") = Rep_TMP  &  "\WMF.tmp"
WshEnv("PAC7FH") = Rep_TMP  &  "\WHF.tmp"
Return = WshShell.Run("BVPAN250.exe" , 1, TRUE)
If Return = 12 Then
Call Msg_Log (Array("1026" , "PAN250"))
End If
Call Err_Cod(Return , 4 , "PAN250")

Call Msg_Log (Array("1022" , "PAN260"))
'-----------------------------------
WshEnv("PAC7OF") = Rep_BASE &  "\FO"
WshEnv("PAC7FO") = Rep_BASE &  "\FO.NEW"
Return = WshShell.Run("BVPAN260.exe" , 1, TRUE)
If Return = 12 Then
Call Msg_Log (Array("1026" , "PAN260"))
End If
Call Err_Cod(Return , 0 , "PAN260")

Call Msg_Log (Array("1022" , "PAN215"))
'-----------------------------------
WshEnv("PAC7HF") = Rep_BASE &  "\FH.NEW"
WshEnv("PAC7FR") = Rep_BASE &  "\FR.NEW"
WshEnv("PAC7FH") = Rep_TMP  &  "\WFH.tmp"
Return = WshShell.Run("BVPAN215.exe" , 1, TRUE)
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 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 used (accepted or rejected) during a thorough 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 correction of the problem.

RESULT

The procedure prints the entry points and the search criteria.

IPFQ : User Input

One 'S' line per criteria selection (optional).

Pos.	Len.	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

Pos.	Len.	Value	Meaning
		'*'	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		Type	Label
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database Index file
PACGGR		Input	Administration Database Data file
PACGGU		Input	Administration Database Users
PAC7AR		Input	Development Database Data file
PAC7AN		Input	Development Database Index file
PAC7FQ		Input	Criteria impacted during analysis
PAC7MB		Input	User input
PAC7FH		Output	Search criteria file
PAC7IX		Report	Output report

PRINTING OF IMPACTED CRITERIA: PAN220

Code		Type	Label
PAC7AE		Input	Error messages
PAC7HF		Input	Sorted entry points or criteria
PAC7IL		Report	List of entry points and criteria

Return codes:

- 0 : OK
- 12 : System error

IPFQ : Execution Script

```
REM * -------------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -------------------------------------------------------
REM *            - IMPACT ANALYSIS:  GROUP FIELDS ANALYSIS -
REM *
REM * -------------------------------------------------------
REM *
REM * THE IPFQ PROCEDURE PRINTS ALL THE ENTRY POINTS AND
REM * IMPACT SEARCH CRITERIA USED (ACCEPTED OR REJECTED)
REM * DURING A THOROUGH IMPACT ANALYSIS.
REM * ALL THE CRITERIA AND ENTRY POINTS ARE STORED IN THE FQ
REM * FILE.
REM * PROCEDURE, IF THEY ARE OF THE GROUP TYPE.
```

```
REM * -------------------------------------------------
REM *
<job id=IPFQ>

<script language="VBScript">
MyProc = "IPFQ"
</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" , "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"
WshEnv("PAC7FH") = Rep_BASE & "\FH"
WshEnv("PAC7FQ") = Rep_BASE & "\FQ"
WshEnv("PAC7MB") = Fic_Input
WshEnv("PAC7IX") = Rep_USR & "\IPFQIX240.txt"
Return = WshShell.Run("BVPAN240.exe" , 1, TRUE)
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("PAC7HF") = Rep_BASE & "\FH"
WshEnv("PAC7IL") = Rep_USR & "\IPFQIL220.txt"
Return = WshShell.Run("BVPAN220.exe" , 1, TRUE)
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>
```

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.

USER INPUT

No user input is required for the execution of the IPEP procedure.

PRINTOUTS

Printout of entry points.

IPEP : Description of Steps

PRINTING OUT ENTRY POINTS: PAN220

Code		Type	Label
PAC7AE		Input	Error messages
PAC7HF		Input	Sorted entry points file
PAC7IL		Input	List of entry points

.Return Codes :

. 0 : OK.

. 12 : System error

IPEP : Execution Script

```
REM * -------------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -------------------------------------------------------
REM *     - IMPACT ANALYSIS:  PRINTING OF ENTRY POINTS -
REM *
REM * -------------------------------------------------------
REM *
REM * THE IPEP PROCEDURE PRODUCES TWO TYPES OF PRINTOUTS.
REM *   1. LIST OF ENTRY POINTS:
REM * THIS LIST IS OBTAINED AFTER THE ISEP PROCEDURE, SINCE
```

```
REM * THIS PROCEDURE SELECTS THE ENTRY POINTS.
REM *   2. LIST OF IMPACT SEARCH CRITERIA:
REM * THIS LIST IS OBTAINED AFTER THE IANA PROCEDURE, SINCE
REM * THIS PROCEDURE SELECTS THE IMPACT SEARCH CRITERIA.
REM * ------------------------------------------------------
REM *
<job id=IPEP>

<script language="VBScript">
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"
WshEnv("PAC7IL") = Rep_USR & "\IPEPIL220.txt"
Return = WshShell.Run("BVPAN220.exe" , 1, TRUE)
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.

Possible reports produced by IPIA are the following:

- 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 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 uses of each of the character strings searched by the ISOS procedure.

  Report requested by value '1' in Position 19 of the P-type user input line.
- Operator analysis:

  List of uses of each of the operators searched by the ISOS procedure.

  Report requested by value '1' in Position 20 of the P-type user input line.
- List of entities impacted by entry point:

  List of 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 analysys 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 lowest 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:

Pos.	Len.	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 = DD/MM/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	' '	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 print
13	1	' '	Does not inhibit lines indirectly impacted)
		'1'	General option: Inhibits 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	Number of the wanted level (IANA iteration)
17	2	pp	Number of lines printed per page
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 Constant-analysis result
		'1'	Constant-analysis result
24	1	' '	No Result of group fields
24	1	'1'	Result of group fields
25	10		Selection of generated transactions
		Blank	Selection of all entities

Pos.	Len.	Value	Meaning
		other	Requested selection, where possible values (compoundable) are:
		'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
35	1	'$' ' '	User Entities No Result with ISOS transactions
		'1'	Result with ISOS transactions

IPIA : Description of Steps

PRINTING OF IMPACT RESULTS: PAN270

Code		Type	Label
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database Index file
PACGGR		Input	Administration Database Data file
PACGGU		Input	Administration Database Users
PAC7AR		Input	Development Database Data file
PAC7FO		Input	Impact results
PAC7MB		Input	User input
PAC7BM		Output	Converted user input
PAC7GY		Output	PAF transactions for UPDP (length=310)
PAC7MV		Output	Batch transactions for printing (length=80)
PAC7IF		Report	Analysis results

Return Codes :

- 0 : OK
- 12 : System error

PRINTING OF GENERATED TRANSACTIONS: PAN280

Code		Type	Label
PAC7AE		Input	Error messages
PAC7BM		Input	User input
PAC7MV		Input	Generated batch transactions
PAC7VM		Output	Selected batch transactions (length=80)
PAC7IE		Report	List of transactions per Library

Return Codes :

- 0 : OK
- 12 : System error

IPIA : Execution Script

```
REM * -----------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -----------------------------------------------------
REM *        - IMPACT ANALYSIS : PRINTING OF RESULTS -
REM *
REM * -----------------------------------------------------
REM *
REM * THE IPIA PROCEDURE IS USED TO PRINT
REM * REPORTS ON THE ANALYSIS RESULTS
REM * AND TO FORMAT THESE RESULTS IN
REM * BATCH UPDATE TRANSACTIONS.
REM * -----------------------------------------------------
REM *
<job id=IPIA>

<script language="VBScript">
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"
WshEnv("PAC7BM") = Rep_TMP & "\MB"
WshEnv("PAC7MV") = Rep_INPUT & "\MVIPIA.bvp"
WshEnv("PAC7IF") = Rep_USR & "\IPIAIF270.txt"
WshEnv("PAC7GY") = Rep_TMP & "\WGY.tmp"
Return = WshShell.Run("BVPAN270.exe" , 1, TRUE)
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"
WshEnv("PAC7BM") = Rep_TMP & "\MB"
WshEnv("PAC7MV") = Rep_INPUT & "\MVIPIA.bvp"
WshEnv("PAC7VM") = Rep_INPUT & "\VMIPIA.bvp"
WshEnv("PAC7IE") = Rep_USR & "\IPIAIE280.txt"
Return = WshShell.Run("BVPAN280.exe" , 1, TRUE)
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 supplied for users of the WorkStation and the SSADM Pacdesign application Design Methodology.

It checks the validity and the consistency of the entities that have been uploaded by the user from the workstation to the specifications Database.

NOTE: The SSADM methodology and the features of the SADM procedure are available only in English.

For further information, refer to the ″Pacdesign″ manual.

EXECUTION CONDITIONS

None.

SADM : User Input

One ′*′ line for library access:

Pos.	Len.	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 version if test session
37	25	.....	Reserved IMS: request identifier (cf. IMS BATCH PAF)

Print request lines:

Pos.	Len.	Value	Meaning
2	1	′T′	Line code
3	1		Code for Report to be printed
		′V′	Validation of SSADM Entities
		′1′	Cross-boundaries Data flows wihin 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		Type	Label
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database Index file
PACGGR		Input	Administration Database Data file
PACGGU		Input	Administration Database Users
PAC7AR		Input	Development Database Data file
PAC7AN		Input	Development Database Index file
PAC7MB		Input	User input
SYSPAF		Input/Output	Standard PAF indexed file
PAC7EJ		Report	List of checked SSADM entities
PAC7DD		Report	List of errors

SADM : Execution Script

```
REM * -----------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -----------------------------------------------------
REM *           - PACDESIGN SSADM INTEGRITY CHECKING -
REM *
REM * -----------------------------------------------------
REM *
REM * THIS PROCEDURE IS SUPPLIED FOR USERS OF THE WORKSTATION
REM * AND THE SSADM PACDESIGN APPLICATION DESIGN METHODOLOGY.
REM *
REM * INPUT :
REM *  -  USER IDENTIFICATION LINE (REQUIRED)
REM *     COL  2 : '*'
REM *     COL  3 : USERIDXX
REM *     COL 11 : PASSWORD
REM *     COL 19 : (BBB)    LIBRARY CODE
REM *     COL 22 : (4 N)    SESSION NUMBER
REM *     COL 26 : (1 CAR.) SESSION VERSION
REM *     COL 37   (25 CAR.) RESERVED IMS
REM *  -  COMMAND LINE :
REM *  COL  2  : 'T'        LINE CODE
REM *  COL  3  :  CODE FOR REPORT TO BE PRINTED
REM *                 'V' : VALIDATION OF SSADM ENTITIES
REM *                 '1' : CROSS-BOUNDARIES DATA FLOWS
REM *                       WITHIN A DFD
REM *                 '2' : OPERATIONAL MASTERS WITHIN A DSD
REM *                 '3' : ALL ENTITIES WITH THEIR ATTRIBUTES
REM *  COL  4  :  (6 CAR.) ENTITY CODE
REM *                       (REQUIRED FOR '1' OR '2')
REM *
REM * -----------------------------------------------------
REM *
```

```
<job id=SADM>

<script language="VBScript">
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"
WshEnv("PAC7DD") = Rep_USR & "\SADMDDM10.txt"
WshEnv("PAC7EJ") = Rep_USR & "\SADMEJM10.txt"
WshEnv("PAC7MB") = Fic_Input
WshEnv("SYSPAF") = Rep_TMP & "\SYSPAF.tmp"
Return = WshShell.Run("BVPADM10.exe" , 1, TRUE)
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 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.
- It establishes different hierarchical lists of certain entities of the Database.

NOTE: The YSM Methodology and the procedure functionalities exist only in English.

For complete details, refer to the ″Pacdesign″ manual.

EXECUTION CONDITIONS

None.

YSMC : User Input - Printed Report

USER INPUT

One '*'-line for library access (required):

Pos.	Len.	Value	Meaning
2	1	'*'	Line code
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):

Pos.	Len.	Value	Meaning
2	1	'T'	Line code
3	1		Code of report to be printed
		'W'	'Validation of YSM entities'

PRC entity control request lines (optional):

Pos.	Len.	Value	Meaning
2	1	'T'	Line code
3	1		Code of report to be printed
		'Y'	'Inter process consistency checking'
4	6	eeeeee	Entity code (PRC)

Print-request lines (optional):

Pos.	Len.	Value	Meaning
2	1	'T'	Line code
3	1		Code of report to be printed
		'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'
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

YSM METHOD INTEGRITY CHECKING: PYSMCC

Code		Type	Label
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database Index file
PACGGR		Input	Administration Database Data file
PACGGU		Input	Administration Database Users
PAC7AR		Input	Development Database Data file
PAC7AN		Input	Development Database Index file
PAC7MB		Input	User input
SYSPAF		Input/Output	Standard PAF indexed file
PAC7EJ		Report	Integrity checking lists
PAC7EI		Report	Validation reports
PAC7DD		Report	Error list

INTER-PROCESS CONSISTENCY: PYSMC3

Code		Type	Label
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database Index file
PACGGR		Input	Administration Database Data file
PACGGU		Input	Administration Database Users
PAC7AR		Input	Development Database Data file
PAC7AN		Input	Development Database Index file
PAC7MB		Input	User input
SYSPAF		Input/Output	Standard PAF indexed file
PAC7EJ		Report	Integrity checking Lists

LIST OF RELATIONSHIPS AND REPORTS: PYSMC2

Code		Type	Label
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database Index file
PACGGR		Input	Administration Database Data file

Code		Type	Label
PACGGU		Input	Administration Database Users
PAC7AR		Input	Development Database Data file
PAC7AN		Input	Development Database Index file
PAC7MB		Input	User input
SYSPAF		Input/Output	Standard PAF indexed file
PAC7EJ		Report	Integrity checking lists

YSMC : Execution Script

```
REM * -----------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -----------------------------------------------------
REM *          - PACDESIGN YSM INTEGRITY CHECKING -
REM *
REM * -----------------------------------------------------
REM *
REM * THIS PROCEDURE IS SUPPLIED FOR USERS OF THE WORKSTATION
REM * AND THE YSM PACDESIGN APPLICATION METHODOLOGY.
REM *
REM *
REM *     IT CHECKS THE VALIDITY AND THE INTEGRITY OF THE
REM * ENTITIES UPLOADED FROM THE WORKSTATION TO THE HOST
REM * SPECIFICATIONS DICTIONARY BY THE USER.
REM *     IT CHECKS THE CONSISTENCY BETWEEN A DATA FLOW
REM * DIAGRAM AND ITS PARENT DIAGRAM.(PRC)
REM *     IT ESTABLISHES DIFFERENT HIERARCHICAL LISTS OF
REM * CERTAIN ENTITIES OF THE DATABASE.
REM * -----------------------------------------------------
REM *
<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"
WshEnv("PAC7DD") = Rep_USR & "\YSMCDDMCC.txt"
WshEnv("PAC7EI") = Rep_USR & "\YSMCEIMCC.txt"
WshEnv("PAC7EJ") = Rep_USR & "\YSMCEJMCC.txt"
WshEnv("PAC7MB") = Fic_Input
WshEnv("SYSPAF") = Rep_USR & "\SYSPAF"
Return = WshShell.Run("BVPYSMCC.exe" , 1, TRUE)
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"
WshEnv("PAC7EJ") = Rep_USR & "\YSMCEJMC3.txt"
WshEnv("PAC7MB") = Fic_Input
WshEnv("SYSPAF") = Rep_USR & "\SYSPAF"
Return = WshShell.Run("BVPYSMC3.exe" , 1, TRUE)
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"
WshEnv("PAC7EJ") = Rep_USR & "\YSMCEJMC2.txt"
WshEnv("PAC7MB") = Fic_Input
WshEnv("SYSPAF") = Rep_TMP & "\SYSPAF.tmp"
Return = WshShell.Run("BVPYSMC2.exe" , 1, TRUE)
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>
```