VisualAge Pacbase

The Administrator's Procedures Windows 2000 or NT Server

Version 3.0

Note

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

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.

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. OVERVIEW

Presentation of the manual

This manual contains the descriptions of all the batch procedures used by a VisualAge Pacbase Database Administrator.

These procedures are related mainly to the following operations fields:

- Management of the administration Database,
- Administration of the development Databases,
- Manager's utilities,
- Analysis of activity and quality control,
- Management of versions.

Presentation 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

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 authorization

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 endings

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.

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 !
```

Definition and submission of procedures

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.

Server start-up

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.

3270 Emulator start-up

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. Administration Database Management

ARCH: Backup for administration

ARCH: Introduction

This procedure backs up the Journal file as a sequential file, and re-initializes it both logically and physically.

Archived transactions do not override those transactions that were previously archived, but rather are added to them.

The archived-transaction file may be purged. Purged transactions may then be saved in another file (PQ).

Previously archived transactions can be purged, if requested. (However, non-archived journal transactions cannot be purged.)

EXECUTION CONDITIONS

On-line access must be closed.

ABNORMAL EXECUTIONS

If an abend occurs before the step that creates the Journal file, the procedure can be restarted as it is once the problem has been solved.

Otherwise, the procedure must be restarted after a modification of the user input in order to specify a re-initialization request without a backup of the Journal file, since it has already been backed up.

ARCH: Inputs - Processes - Results

Batch procedure access authorization option: one '*' line with user code and password.

This procedure includes specific optional input for:

- Purging previously archived transactions that are considered obsolete.
- Signalling the absence of previously archived transactions during input.
- Signalling the unavailability of the Data file (AR) during input.
- Requesting the re-initialization of the transaction file only.

The structure of this input is as follows:

POS.	LEN.	VALUE	MEANING
2	1	'S'	Line code
3	4	nnnn	Session number
7	8	ccyymmdd	OR date up to which the user requests deactivation
15	1	'I'	Absence of previously archived transactions
16	1	'D'	Data file unavailable

POS.	LEN.	VALUE	MEANING
17	1	'J'	Re-initialization without backup, the transactions already archived are NOT retrieved on output.

The session number and the date are independent of each other. They are ignored if it is indicated that there are no input transactions (refer to Section Recommendations).

The unavailability of the Data file is to be indicated only when this file has been physically deleted. (See Section Recommendations below.)

A request to re-initialize without archiving is necessary when the Journal file is physically deleted.

Note:

In this case, the transactions which were already archived are not copied on to the transaction output file. (If this input file is automatically catalogued by the operating system, the transactions already archived may be lost unless the file is uncatalogued).

In an error occurs on one of the options, a message is printed and the archive is generated using the default options.

Recommentations

If there is no user input, this procedure can only be executed if the Database is in a consistent state, and if the archived transaction file is correctly formatted.

When the Database needs to be restored after an abend or a system failure, some information in the Specifications Dictionary is sometimes lost, thus preventing the execution of the backup and restauration procedures.

In this case, AND IN THIS CASE ONLY, columns 15 to 17 of the user input are to be used as follows:

- If the Data file (AR) is lost or has been flagged as 'inconsistent', a 'D' in column 16 means that the backup procedure will not take the Data file (AR) into account. However, the restoration procedure must be executed afterward, since under these conditions, the backup procedure leaves the database in an inconsistent state.
- If the Journal file (AJ) is lost or destroyed, a 'J' must be entered in column 17. As a result, the backup procedure formats an empty Journal file. The restoration procedure may then be executed (not compulsory). In this case, the content of the journal file (AJ) is lost.
- If the transactions Back-up file is lost or destroyed, an 'I' must be entered in column 15. As a result, the ARCH procedure reformats a new sequential backup file of (archived) transactions.

If one of these columns is accidentally set, and if the ARCH procedure is executed while the Database is in a consistent state, the consequences are:

- 'I' in col. 15: Previously archived transactions are lost. All transactions can be recovered by concatenating (-1) and (0) to obtain (+1).

- 'D' in col. 16: The ARCH procedure must be re-executed BEFORE any update. If an update is subsequently performed, the Database will be lost, and will have to be restored completely.
- 'J' in col. 17: The contents of the Journal file are definitely lost. The output Journal file, (+1 version in the case of generation data files), is created empty.

Printed output

This procedure prints a report stating the number of archived transactions and, if applicable, the number of records that have been 'purged'.

Results

Once this procedure is executed, a sequential file containing all archived transactions is obtained.

The Journal file which displays transactions on-line is re-initialized.

It is also possible to store on another file all transactions that have been purged.

Note:

This procedure does not increment the session number.

ARCH: Description of Steps

Archival of journal file: PTU300

This step:

- writes obsolete transactions to be purged on to a special file, if the purge is requested in user input.
- positions a flag in the Data file indicating the journal archive.
- updates the file of archived transactions.

Code		Type	Label
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database index
PACGGR		Input	Administration Database data
PACGGU		Input	Administration Database users
PAC7JP		Input	Previously archived transactions
PAC7AJ		Input	Journal to reinitialize from administration database
PAC7MB		Input	User transaction
PAC7BM		Output	User transaction
PAC7AR		Input/Output	Development Database data
PAC7PJ		Output	Archived update transactions
PAC7PQ		Output	Desactivated transactions (length=170) : modify file name to save desactivated transactions
PAC7EU		Report	Output report
PAC7DD		Report	Batch procedures authorization option

Return codes:

- 0: No error detected on the files
- 4: Erroneous record of Journal file
- 8: No access authorization for batch procedures
- 12: Input-output error on a file

Re-initialization of the Journal file: PTU320

This step executes two types of operations:

- Creates the first record in the Journal file,
- Re-initializes the data file flag with the Journal file's address.

Code		Type	Label
PAC7BM		Input	User transaction
PAC7AR		Input/Output	Administration Database data
PAC7AE		Input	Error messages
PAC7AJ		Output	Journal file to re-initialize
PAC7EU		Report	Review of re-initialization

Return codes:

- 0: No error detected
- 8: Invalid database

ARCH: Execution Script

```
REM * -----------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -----------------------------------------------------
REM *             - ARCHIVAL OF THE JOURNAL -
REM *
REM * -----------------------------------------------------
REM *
REM * INPUT       : COMMAND FOR DEACTIVATION OF ARCHIVED
REM *               TRANSACTION
REM * COL 2       : 'S'
REM * COL 3 TO  6 : SESSION NUMBER
REM * COL 7 TO 14 : DATE (CCYYMMDD)
REM * COL 15      : ' ' PRESENCE OF ARCHIVED TRANSACTION FILE
REM *             : 'I' ABSENCE OF ARCHIVED TRANSACTION FILE
REM * COL 16      : ' ' PRESENCE OF DATA FILE (AR)
REM *             : 'D' ABSENCE OF DATA FILE (AR)
REM * COL 17      : ' ' ARCHIVAL AND REINITIALIZATION
REM *             : 'J' REINITIALIZATION WITHOUT ARCHIVAL
REM *
REM * IN THE ABSENCE OF INPUT (OR ERROR ON A COMMAND PARAM.)
REM * NO DEACTIVATION WILL TAKE PLACE, HOWEVER ARCHIVAL AND
REM * REINITIALIZATION WILL BE EXECUTED NORMALLY.
REM *
REM * TRANSACTIONS WHOSE SESSION (DATE) IS PRIOR OR EQUAL TO
REM * THE SESSION (DATE) INDICATED ARE NOT KEPT. THEY ARE
REM * RECOVERED IN THE FILE OF DEACTIVATED TRANSACTION.
REM *
REM * -----------------------------------------------------
<job id=ARCH>

<script language="VBScript">
MyProc = "ARCH"
</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" , "PTU300"))
'------------------------------------
WshEnv("PAC7MB") = Fic_Input
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PAC7AJ") = Rep_JOURNAL & "\AJ"
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7JP") = Rep_SAVE & "\PJ"
WshEnv("PAC7PJ") = Rep_SAVE & "\PJ.new"
WshEnv("PAC7PQ") = "NUL"
WshEnv("PAC7BM") = Rep_TMP & "\WBM.tmp"
WshEnv("PAC7EU") = Rep_USR & "\ARCHEU300.txt"
WshEnv("PAC7DD") = Rep_USR & "\ARCHDD300.txt"
Return = WshShell.Run("BVPTU300.exe" , 1, TRUE)
If Return = 8 Then
Call Msg_Log (Array("1032"))
End If
If Return = 12 Then
Call Msg_Log (Array("1026"))
End If
Call Err_Cod(Return , 0 , "PTU300")



Call Msg_Log (Array("1022" , "PTU320"))
'------------------------------------
WshEnv("PAC7EU") = Rep_USR & "\ARCHEU320.txt"
Return = WshShell.Run("BVPTU320.EXE" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTU320")

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

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

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

</script>
</job>
```

PACS: Backup for Administration

PACS: Introduction

The purpose of this procedure is to save the main files of the administration Database with the 'PE' sequential file format.

Backup is performed on the following files:

- the data file (AR),
- the index file (AN),

- the extension data file (GY).

Execution conditions

On-line access must be prohibited.

Abnormal execution

Refer to sub-chapter ″Abnormal endings′, Chapter ″Overview″.

The main reason for an abend is that on-line environment is still open to transactions.

The procedure can therefore be restarted once the on-line environment is closed.

Archival and Backup linking

If the backup procedure is preceded by a Journal archival procedure (ARAD), its execution may be conditioned by the return code PTU320 ARAD step:

- 0: No error detected
- 8: Invalid database

Printed report

Once the procedure is executed, the following reports are printed:

- A report containing the number of records saved in each file and the session number,
- Two optional reports:
  - a statistical report with the number of records per library and per line-type,
  - a report listing database limits reached.

PACS: Inputs - Processes - Results

Pos.	Len.	Value	Meaning
2	1	'*'	Line code
3	8	uuuuuuuu	User code
11	8	pppppppp	Password
29	4	'SAVE'	Function code

PACS: Description of Steps

Formating sequential image: PTU520

Code		Type	Label
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database index
PACGGR		Input	Administration Database data
PACGGU		Input	Administration Database users
PAC7AR		Input	Administration Database data
PAC7AY		Input	Administration Database extension data

Code		Type	Label
PAC7AN		Input	Administration Database index
PAC7MB		Input	Update transactions
PAC7PC		Sortie	Sequential image of the administration Database
PAC7RP		Output	Sequential image of data (length=153) (should be able to contain all the data)
PAC7NA		Output	Sequential image of indexes (length=59) (should be able to contain all the indexes)
PAC7NB		Output	Image of desorted indexes (length=59)
PAC7RY		Output	Sequential image of long data (length=1,019)
PAC7RQ		Output	Intermediary storage (1 record, length=153)
PAC7EV		Report	User transactions list
PAC7EU		Report	Status of network before and after
PAC7EW		Report	Backup report
PAC7DD		Report	Abend output report

Return code:

- 8 : Inconsistency of the database or no authorization for batch procedure

Response to return code:

If the return code is greater than 2, the resulting backup is deleted by the next step in the procedure and a restoration must be performed using the last valid backup.

If there is no other backup to restore the database, the user should first examin the problem with the support team of the product, then, the inconsistent database should be saved by the same procedure with the backup deletion step inactive. The resulting backup contains only data and can only be used after running the reorganization procedure.

PACS: Execution Script

```
REM * -----------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -----------------------------------------------------
REM *                 - BACKUP OF THE DATABASE -
REM *
REM * -----------------------------------------------------
REM *
<job id=PACS>

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

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

<script language="VBScript">

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


Dim Ret520
Ret520 = 0
```

```
Call Msg_Log (Array("1022" , "PTU520"))
'-----------------------------------
WshEnv("PAC7MB") = Fic_Input
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7AY") = Rep_BASE & "\AY"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("SEMLOCK") = Rep_BASE & "\LO"
WshEnv("SEMADMIN") = Rep_ABASE & "\LO"
WshEnv("PAC7PC") = Rep_SAVE & "\PC.NEW"
WshEnv("PAC7PD") = Rep_SAVE & "\PCI.NEW"
WshEnv("PAC7PY") = Rep_SAVE & "\PCY.NEW"
WshEnv("PAC7NA") = Rep_TMP & "\WNA.tmp"
WshEnv("PAC7NB") = Rep_TMP & "\WNB.tmp"
WshEnv("PAC7RP") = Rep_TMP & "\WRP.tmp"
WshEnv("PAC7RQ") = Rep_TMP & "\WRQ.tmp"
WshEnv("PAC7RY") = Rep_TMP & "\WRY.tmp"
WshEnv("PAC7EU") = Rep_USR & "\PACSEU520.txt"
WshEnv("PAC7DD") = Rep_USR & "\PACSDD520.txt"
WshEnv("PAC7EW") = Rep_USR & "\PACSEW520.txt"
WshEnv("PAC7EV") = Rep_USR & "\PACSEV520.txt"
Return = WshShell.Run("BVPTU520.exe" , 1, TRUE)
Ret520= Return
If Return = 4 then Return = 0 end if

If Ret520 = 2 then
Call Msg_Log (Array("1022" , "PTU530"))
'-----------------------------------
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PAC7PC") = Rep_SAVE & "\PC.NEW"
WshEnv("PAC7PD") = Rep_SAVE & "\PCI.NEW"
WshEnv("PAC7PY") = Rep_SAVE & "\PCY.NEW"
WshEnv("PAC7NA") = Rep_TMP & "\WNA.tmp"
WshEnv("PAC7NB") = Rep_TMP & "\WNB.tmp"
WshEnv("PAC7RP") = Rep_TMP & "\WRP.tmp"
WshEnv("PAC7RQ") = Rep_TMP & "\WRQ.tmp"
WshEnv("PAC7RY") = Rep_TMP & "\WRY.tmp"
Return = WshShell.Run("BVPTU530.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTU530")
Else
If Return = 8 Then
Call Msg_Log (Array("1051"))
End If
Call Err_Cod(Return , 4 , "PTU520")
End If

If Ret520 <> 4 then
Call Msg_Log (Array("1022" , "BACKUP"))
'-----------------------------------
Call Turnover(Rep_SAVE & "\PC")
Call Turnover(Rep_SAVE & "\PCI")
Call Turnover(Rep_SAVE & "\PCY")
End If


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

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

```
Wscript.Quit (Return)

</script>
</job>
```

REOR: Reorganization for Administration

REOR: Introduction

The Database Reorganization procedure optimizes Database accesses by accounting for each deletion, and sorting the data again according to the most frequent access order.

It uses the administration Database backup file (PE) and rebuilt a sequential image. This resulting image file must then be restored via the REST procedure.

The operating principle of this procedure is to rebuild the different indexes associated with all data using the 'image' of each Data Element. It makes the best of the system performance features since it separates historical (frozen) sessions from the current session and sorts the data in the order of the most frequent access. This makes it possible to achieve a significant reduction of the number of indexes and data items.

This procedure may be used in two cases:

- When part of the data was deleted because of a malfun tion or system failure, and no other procedure can be used (in particular, deletion of the Index file).
- When the administrator wants to purge the entities not used in the database.

This procedure should be executed only on an exceptional basis, because of the special conditions concerning its use.

Execution conditions

The administration Database may remain open during the reorganization since the procedure operates on sequential images of the database.

Updates executed after the back-up file used for reorganization has been built will be retrievable while the reorganized database is being restored.

Abends

Refer to sub-chapter 'Abnormal endings', Chapter 'Overview'.

As specified in the recommendations below, it is advisable to keep all temporary files after each step.

If one of the steps abends, the procedure can be restarted at the step level, but not at the procedure level.

REOR: Inputs - Processes - Results

Batch procedure access authorization option: one '*' line with user code and password.

Specific user input for the procedure (optional), specifying a printed copy of the list of index of the reorganization procedure.

Pos.	Len.	Value	Meaning
2	1	'D'	Printed copy of the list of index of the REOR procedure
3	1	' '	No report of copies of index
		'1'	Report of copies of index

When the system finds an input error, it generates an error message and the procedure is not executed.

Estimating file size

The maximum sizes used during this procedure are based on the sizes of the files in the database before reorganization. The report printed by the preceding procedure provides all the relevant data:

- NI = number of index file records.
- ND = number of data file records MINUS number of gaps.
- NC = number of primary records on the data file.
- NH = number of 'frozen' (historical account) records from the data file (NH = ND - NC).

These symbols are also detailed in the presentation of each of the files for this procedure.

Printed output

This procedure prints a report listing errors encountered during reorganization, and statistics on the contents of the database.

It also prints reports with the staement 'Internal report' reserving their use to the product support team in case of problem.

Results

The output of this procedure is a reorganized sequential image of the database (where purges may have been performed). It does not contain gaps. Gaps can be added by the database restoration procedure.

Important recommendations

The Reorganization procedure presents a number of characteristics which the user should be aware of:

- The step that rebuilds the Index file uses a large amount of CPU time.
- If the database contains a large amount of data, it is recommended to catalog the temporary files, or to use tape files to obtain the checkpoints in case of an abend in one of the steps.
- If files are transferred onto tape it is preferable to check on the initial blocking factors.
- The space allocated to the sortworks should also be calculated with care.

REOR: Description of Steps

Input check: PTU2CL

This step checks all user inputs and set a return code if there is no error detected.

Code		Type	Label
PAC7AE		Input	Error labels
PACGGN		Input	Administration database index
PACGGR		Input	Administration database data
PACGGU		Input	Administration database users
PAC7PC		Input	Sequential image of the administration Database
PAC7MB		Input	Input working file
PAC7BM		Output	Records formatted
PAC7PU		Output	transactions to purge entities (length=44)
PAC7EE		Report	Report on ckecks
PAC7DD		Report	Batch procedure authorization option

Retrun codes:

- 0: OK
- 4: Error on user input
- 8: No batch procedure authorization

Data retrieval: PTU200

This step selects 'data' type information in the initial sequential image and then formats the key of each record selected for the subsequent sort.

Code		Type	Label
PAC7AE		Input	Error messages
PAC7BM		Input	User transactions
PAC7PC		Input	Administration Database sequential image
PAC7PR		Output	Formatted records (length=176 size = ND)
PAC7NX		Output	Long data
PAC7NY		Output	Unformatted data
PAC7AU		Output	PR image (length=153)
PAC7PY		Output	PY image (length=1,036)
PAC7EE		Report	Retrieval statistics output report

Tri ASCII : PTU205

Code		Type	Label
PAC7PR		Entrée	Sorted records
PAC7RP		Sortie	ASCII sorted records

The SORT programs requires disk space equivalent to twice the size of the file to be sorted.

Purge: PTU210

This step formats the records.

Code		Type	Label
PAC7AE		Input	Error messages
PAC7PR		Input	Sorted records
PAC7PU		Input	Entity records to be purged
PAC7BM		Input	User transactions
PAC7QS		Output	Purged records (length=176, size= ND)
PAC7UM		Output	Macro-strucrure call lines (length=176)
PAC7EE		Report	Library and session purge report
PAC7EK		Report	Entity-purge report
PAC7EB		Report	Technical report

Return codes:

- 0: OK
- 8: Overload of capacity

The steps that follow are executed only if the return code for the purge step is zero.

Rebuilding index: PTU220

This step reconstitutes indexes using the data.

Code		Type	Label
PAC7AE		Input	Error messages file
PAC7BM		Input	User transactions
PAC7UR		Input	Purged data
PAC7NX		Input	Long data
PAC7UM		Input	Macro-structure call lines
PAC7PA		Output	Data from frozen session (length=153 size=NH)
PAC7PB		Output	Data from the current session (length=153 size=NC)
PAC7PC		Output	First data record (length=153)
PAC7AN		Output	Temporary index file (length=60 size=NI)
PAC7MR		Input/Output	Macro-structure call lines
PAC7EE		Report	Report of index building

Tri ASCII : PTU205

Code		Type	Label
PAC7AN		Input	Sorted index
PAC7NA		Output	ASCII Sorted index

The SORT programs requires disk space equivalent to twice the size of the file to be sorted.

Extension data processing: PTU226

Code		Type	Label
PAC7NY		Input	Unformatted data
PAC7PA		Input	Data from frozen session
PAC7PB		Input	Data from the current session
PAC7PC		Input	First data record
PAC7QA		Output	Data from frozen sessions (length=153)
PAC7QB		Output	Data from the current session (length=153)
PAC7QC		Output	First data record (length=153)
PAC7QY		Output	Long data (length=1018)

Merge: PTU240

This step rebuilds the final sequential image using the temporary files produced by the previous step.

Code		Type	Label
PAC7AE		Input	Error messages
PAC7AN		Input	Sorted index
PAC7AU		Input	PR image
PAC7BM		Input	User transactions
PAC7PA		Input	Data from frozen sessions
PAC7PB		Input	Data from the current session
PAC7PC		Input	First data record
PAC7QY		Input	Extension data
PAC7CP		Output	Sequential image of the administration Database
PAC7IE		Report	Building the logical database

REOR: Execution Script

```
REM * -------------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -------------------------------------------------------
REM *                - REORGANIZATION OF THE DATABASE -
REM *
REM * -------------------------------------------------------
REM *
REM * THE REOR PROCEDURE MAY BE USED IN TWO CASES:
REM * . WHEN PART OF THE DATA WAS DELETED BECAUSE OF A MAL-
REM *  FUNCTION OR SYSTEM FAILURE, AND NO OTHER PROCEDURE CAN
REM *  BE USED (IN PARTICULAR, DELETION OF THE AN INDEX FILE)
REM * . WHEN THE DATABASE IS TO BE PURGED OF THE FOLLOWING:
REM *   - OBSOLETE LIBRARIES AND/OR SESSIONS;
REM *   - ENTITIES NOT USED IN THE DATABASE;
REM *
REM * -------------------------------------------------------
REM *
<job id=REOR>

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

```
</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" , "PTU2CL"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7BM") = Rep_TMP & "\WBM.tmp"
WshEnv("PAC7PU") = Rep_TMP & "\WPU.tmp"
WshEnv("PAC7DD") = Rep_USR & "\REORDD2CL.txt"
WshEnv("PAC7EE") = Rep_USR & "\REOREE2CL.txt"
WshEnv("PAC7MB") = Fic_Input
WshEnv("PAC7PC") = Rep_SAVE & "\PC"
Return = WshShell.Run("BVPTU2CL.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTU2CL")

Call Msg_Log (Array("1022" , "PTU200"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7PC") = Rep_SAVE & "\PC"
WshEnv("PAC7PY") = "NUL"
' WshEnv("PAC7PY") = Rep_SAVE & "\PCY"
WshEnv("PAC7BM") = Rep_TMP & "\WBM.tmp"
WshEnv("PAC7PR") = Rep_TMP & "\WPR.tmp"
WshEnv("PAC7NX") = Rep_TMP & "\WNX.tmp"
WshEnv("PAC7NY") = Rep_TMP & "\WNY.tmp"
WshEnv("PAC7AU") = Rep_TMP & "\WAU.tmp"
WshEnv("PAC7EE") = Rep_USR & "\REOREE200.txt"
Return = WshShell.Run("BVPTU200.EXE" , 1, TRUE)
If Return = 8 Then
Call Msg_Log (Array("1027"))
End If
Call Err_Cod(Return , 0 , "PTU200")

Call Msg_Log (Array("1022" , "PTU205"))
'-----------------------------------
WshEnv("PAC7PR") = Rep_TMP & "\WPR.tmp"
WshEnv("PAC7RP") = Rep_TMP & "\WRP.tmp"
Return = WshShell.Run("BVPTU205.EXE" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTU205")
'OK :
Call DelFile (Rep_TMP & "\WPR.tmp")

Call Msg_Log (Array("1022" , "PTU210"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7EB") = Rep_USR & "\REOREB210.txt"
WshEnv("PAC7EE") = Rep_USR & "\REOREE210.txt"
WshEnv("PAC7EK") = Rep_USR & "\REOREK210.txt"
WshEnv("PAC7BM") = Rep_TMP & "\WBM.tmp"
WshEnv("PAC7PR") = Rep_TMP & "\WRP.tmp"
WshEnv("PAC7PU") = Rep_TMP & "\WPU.tmp"
WshEnv("PAC7QS") = Rep_TMP & "\WQS.tmp"
WshEnv("PAC7UM") = Rep_TMP & "\WUM.tmp"
Return = WshShell.Run("BVPTU210.EXE" , 1, TRUE)
If Return = 4 Then
Call Msg_Log (Array("1057"))
End If
If Return = 8 Then
Call Msg_Log (Array("1056"))
```

```
End If
Call Err_Cod(Return , 0 , "PTU210")
'OK :
Call DelFile (Rep_TMP & "\WRP.tmp")
Call DelFile (Rep_TMP & "\WPU.tmp")

Call Msg_Log (Array("1022" , "PTU220"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7BM") = Rep_TMP & "\WBM.tmp"
WshEnv("PAC7AN") = Rep_TMP & "\WAN.tmp"
WshEnv("PAC7MR") = Rep_TMP & "\WMR.tmp"
WshEnv("PAC7NX") = Rep_TMP & "\WNX.tmp"
WshEnv("PAC7PA") = Rep_TMP & "\WPA.tmp"
WshEnv("PAC7PB") = Rep_TMP & "\WPB.tmp"
WshEnv("PAC7PC") = Rep_TMP & "\WPC.tmp"
WshEnv("PAC7UM") = Rep_TMP & "\WUM.tmp"
WshEnv("PAC7UR") = Rep_TMP & "\WQS.tmp"
WshEnv("PAC7EE") = Rep_USR & "\REOREE220.txt"
Return = WshShell.Run("BVPTU220.EXE" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTU220")

Call Msg_Log (Array("1024"))
'-----------------------------------
Call DelFile (Rep_TMP & "\WUM.tmp")
Call DelFile (Rep_TMP & "\WQS.tmp")

Call Msg_Log (Array("1022" , "PTU225"))
'-----------------------------------
WshEnv("PAC7AN") = Rep_TMP & "\WAN.tmp"
WshEnv("PAC7NA") = Rep_TMP & "\WNA.tmp"
Return = WshShell.Run("BVPTU225.EXE" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTU225")

Call Msg_Log (Array("1024"))
'-----------------------------------
Call DelFile (Rep_TMP & "\WAN.tmp")

Call Msg_Log (Array("1022" , "PTU226"))
'-----------------------------------
WshEnv("PAC7PA") = Rep_TMP & "\WPA.tmp"
WshEnv("PAC7PB") = Rep_TMP & "\WPB.tmp"
WshEnv("PAC7PC") = Rep_TMP & "\WPC.tmp"
WshEnv("PAC7QA") = Rep_TMP & "\WQA.tmp"
WshEnv("PAC7QB") = Rep_TMP & "\WQB.tmp"
WshEnv("PAC7QC") = Rep_TMP & "\WQC.tmp"
WshEnv("PAC7QY") = Rep_TMP & "\WQY.tmp"
WshEnv("PAC7NY") = Rep_TMP & "\WNY.tmp"
Return = WshShell.Run("BVPTU226.EXE" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTU226")

Call Msg_Log (Array("1024"))
'-----------------------------------
Call DelFile (Rep_TMP & "\WPA.tmp")
Call DelFile (Rep_TMP & "\WPB.tmp")
Call DelFile (Rep_TMP & "\WPC.tmp")

Call Msg_Log (Array("1022" , "PTU240"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_TMP & "\WNA.tmp"
WshEnv("PAC7AU") = Rep_TMP & "\WAU.tmp"
WshEnv("PAC7BM") = Rep_TMP & "\WBM.tmp"
WshEnv("PAC7PA") = Rep_TMP & "\WQA.tmp"
```

```
WshEnv("PAC7PB") = Rep_TMP & "\WQB.tmp"
WshEnv("PAC7PC") = Rep_TMP & "\WQC.tmp"
WshEnv("PAC7QY") = Rep_TMP & "\WQY.tmp"
WshEnv("PAC7CP") = Rep_SAVE & "\PC.new"
WshEnv("PAC7PD") = Rep_SAVE & "\PCI.new"
WshEnv("PAC7PY") = Rep_SAVE & "\PCY.new"
WshEnv("PAC7IE") = Rep_USR & "\REORIE240.txt"
Return = WshShell.Run("BVPTU240.EXE" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTU240")

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

If Return = 0 then
Call Msg_Log (Array("1022" , "BACKUP"))
'-----------------------------------
Call Turnover(Rep_SAVE & "\PC")
Call Turnover(Rep_SAVE & "\PCI")
Call Turnover(Rep_SAVE & "\PCY")
End if

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

</script>
</job>
```

REST: Restoration for Asministration

REST: Introduction

The purpose of this procedure is to rebuild the administration Database with the use of the sequential image resulting from the execution of the backup procedure (PACS SAVE option).

It is also used to retrieve archived transactions via the resulting sequential image and to modify the number of 'gaps' of the Database.

Execution conditions

The administration Database should be closed in the on-line environment.

As the procedure rebuilds the Database, it may be interesting first to change the size of the Database files in expectation of an increase.

These changes must be done in the system paramaters library.

The procedure re-initializes the transactions Journal physically and logically. It is therefore required to execute first the ARCH procedure to backup the Journal.

Abend

Refer to sub-chapter 'Abnormal endings' in Chapter 'Overview'.

Whatever the reason of the abnormal ending, the procedure can be restarted once the problem solved.

REST: Inputs - Processes - Results

A '*' line with user code and password.

The structure of the specific input is described in the chart below:

Pos.	Len.	Value	Meaning
2	1	'Y'	Line code
3	5	nnnnn	Number of gaps as absolute value
8	2	pp	OR number of gaps as % (1)
10	2		Language code (FR, EN, ...)
12	1	'0'	No inhibition of the Journal
		'1'	Journal inhibition (no journalization of updated transactions)
		' '	Retrieval of the last value
14	3	'REC'	If archived transactions retrieved

Notes

Where there is no input, the Database characteristics remain unchanged.

Any field left blank will be filled in with the current options.

If the Journal is inhibited (parameter set to '1'), update transactions are not backed-up in the Journal file. In this case, it is impossible to restore the Database using the recovery of archived transactions ('REC' parameter on user input). It is therefore highly recommended to set this parameterto '0' (which is the default value) so as to avoid restoration problems.

In case of error, invalid parameters are ignored, and the system ensures restoration using the parameter values stored in the sequential image of the Database.

Report results

This procedure prints a report listing the requested options, associated errors, the number of records restored in the Database for each file, the number of gaps, and the options stored in the new Database.

Results

Once the procedure has been executed, the Database is ready to be used in batch or on-line mode.

Notes:

Once this procedure is executed, the current session number is that of the sequential image, or of the most recent transaction if the retrieval of archived transactions has been run.

REST: Description of Steps

Validation of Journal contents: PTU380

This step is executed if the Journal file exists.

Code		Type	Label
PAC7MB		Input	User transactions
PAC7AE		Input	Error messages file
PAC7AJ		Input	Journal file
PAC7EU		Report	(only if the Journal was not archived)

Return code:

- 0: The journal file has been archived

Restoration of the Database: PTU400

This step is executed only if the Journal file has been archived.

Code		Type	Label
PAC7AE		Input	Error labels
PAC7MB		Input	User transactions
PAC7PC		Input	Sequential image of the administration Database
PAC7AR		Output	Administration Database data
PAC7AY		Output	Administration Database extension data
PAC7AN		Output	Administration Database index
PAC7AJ		Output	Administration Database Journal
PAC7PS		Output	Working file (2 records, length=144)
PAC7EU		Report	Restoration report
PAC7DD		Report	Batch procedure authorization option

Database availability - transactions retrieval: PTU420

This step is executed if the Journal file has been archived. It updates the first record of the Data file.

Caution: This step is REQUIRED to obtain a consistent database.

Code		Type	Label
PAC7AE		Input	Error messages
PAC7MB		Input	User transaction
PAC7AR		Input/Output	Administration Database data
PAC7JO		Input	Journal to apply
PAC7PS		Input	Working file
PAC7OJ		Output	Update transcations (length=170)
PAC7EU		Report	Retrival report

Return codes:

- 0: Transactions to retrieve
- 4: No transactions to retrieve

In: case of abnormal ending, the database cannot be updated.

Administration Database update: PACA15

Code		Type	Label
PAC7AR		Output	Administration Database data
PAC7AN		Output	Administration Database index
PAC7AY		Output	Administration Database extension
PAC7AJ		Output	Administration Database Journal
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database index
PACGGR		Input	Administration Database data
PACGGY		Input	Administration Database Extension
PACGGU		Input	Administration Database users
PAC7DC		Input	DSMS file of the development Database
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	Erroneous transactions list (length=132)

The list of transactions proper to a user is preceded by a banner specifying the user code.

REST: Execution Script

```
REM * -----------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -----------------------------------------------------
REM *         - RELOADING RESTORATION OF THE DATABASE -
REM *
REM * -----------------------------------------------------
REM *
REM * INPUT
REM * COL  2     : 'Y'
REM * COL  3-7   : NUMBER OF GAPS IN ABSOLUTE VALUE
REM * COL  8-9   : NUMBER OF GAPS IN PERCENTAGE   ( / BASE )
REM * COL 10-11  : INITIAL LANGUAGE CODE (FR, EN, ..)
REM * COL 12     : '1' INHIBITION OF TRANSACTION LOG
REM * COL 14-16  :'REC' FOR RECOVERY OF ARCHIVED TRANSACTIONS
REM * COL 17-20  : 4 CHARACTERS TO BE DISPLAYED
REM *              ON ALL SCREEN OF THE PRODUCT
REM * COL 21-24  : 'NNNN' MAXIMUM NUMBER OF SEARCH ACCESSES
REM *              TO THE DATABASE(LISTS)-(DEFAULT VALUE:300)
REM * COL 25     : 'U' (DEFAULT VALUE) : IMPLICIT UPDATE
REM *            : 'N' EXPLICIT UPDATE
REM * COL 26-29  : CKECKPOINT FREQUENCY
REM * COL 36-47  : PF-KEYS SIGNIFICATIONS
REM * COL 79     : BACKUP FILES DISPATCH
REM *            : 'N' (DEFAULT VALUE) : NO DISPATCH (1 FILE)
REM *            : 'D' : DISPATCH (3 FILES)
REM *
REM * IN THE ABSENCE OF INPUT, THE RELOAD DOES NOT MODIFY THE
```

```
REM * NUMBER OF EXISTING GAPS, AND OTHER DATA IS UNCHANGED.
REM *
REM * IF THE JOURNAL FILE OF TRANSACTIONS ON DISK (AJ) IS NOT
REM * REINITIALIZED, THE RESTORE CHAIN IS NOT EXECUTED.
REM * IT IS THEREFORE NECESSARY TO EXECUTE THE ARCH PROCEDURE
REM * FIRST.
REM * -------------------------------------------------------
<job id=REST>

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

<script language="VBScript">

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


Dim Ret420
Ret420 = 0

Call Msg_Log (Array("1022" , "PTU010"))
'-------------------------------------
WshEnv("CARTE") = Fic_Input
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7DD") = Rep_USR & "\RESTDD010.txt"
WshEnv("PAC7MB") = Rep_TMP  & "\WMB.tmp"
WshEnv("PAC7PC") = Rep_SAVE & "\PC"
WshEnv("PAC7PD") = Rep_SAVE & "\PCI"
WshEnv("PAC7PY") = Rep_SAVE & "\PCY"
Return = WshShell.Run("BVPTU010.exe" , 1, TRUE)
WshVolEnv("RC") = Return

If Return = 8 Then
Call Msg_Log (Array("1027"))
End If
Call Err_Cod(Return , 0 , "PTU010")

If  FSO.FileExists(Rep_JOURNAL & "\AJ") Then
Call Msg_Log (Array("1022", "PTU380"))
'-------------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AJ") =  Rep_JOURNAL & "\AJ"
WshEnv("PAC7EU") =  Rep_USR & "\RESTEU380.txt"
WshEnv("PAC7MB") = Rep_TMP  & "\WMB.tmp"
Return = WshShell.Run("BVPTU380.EXE" , 1, TRUE)
WshVolEnv("RC") = Return
If Return = 8 Then
Call Msg_Log (Array("1058"))
End If
Call Err_Cod(Return , 0 , "PTU380")
End If

Call Msg_Log (Array("1022" , "PTU400"))
'-------------------------------------
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("PAC7DD") =  Rep_USR & "\RESTDD400.txt"
WshEnv("PAC7EU") =  Rep_USR & "\RESTEU400.txt"
WshEnv("PAC7PC") = Rep_SAVE & "\PC"
WshEnv("PAC7PD") = Rep_SAVE & "\PCI"
WshEnv("PAC7PS") =  Rep_TMP & "\WPS.tmp"
WshEnv("PAC7PY") = Rep_SAVE & "\PCY"
Return = WshShell.Run("BVPTU400.exe" , 1, TRUE)
WshVolEnv("RC") = Return
Call Err_Cod(Return , 0 , "PTU400")

If  FSO.FileExists( Rep_SAVE & "\PJ"  ) Then
Call Msg_Log (Array("1022" , "PTU420"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AR") =  Rep_BASE & "\AR"
WshEnv("PAC7JO") =  Rep_SAVE & "\PJ"
WshEnv("PAC7MB") =  Rep_TMP & "\WMB.tmp"
WshEnv("PAC7OJ") =  Rep_TMP & "\WOJ.tmp"
WshEnv("PAC7PS") =  Rep_TMP & "\WPS.tmp"
WshEnv("PAC7EU") =  Rep_USR & "\RESTEU420.txt"
Return = WshShell.Run("BVPTU420.EXE" , 1, TRUE)
Ret420 = Return

If Return = 4 Then
'No transaction to be retrieved : Normal End
'-----------------------------------------
Call Msg_Log (Array("1059"))
Call Msg_Log (Array("1024"))
Return = 0
WshVolEnv("RC") = Return

Else
WshVolEnv("RC") = Return
Call Err_Cod(Return , 0 , "PTU420")
End If

If Ret420 = 0 then
'Update
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("PACGGY") = Rep_ABASE & "\AY"
WshEnv("PAC7DC") =  Rep_BASE & "\DC"
WshEnv("PAC7IE") =  Rep_USR & "\RESTIEA15.txt"
WshEnv("PAC7IF") =  Rep_USR & "\RESTIFA15.txt"
WshEnv("SEMLOCK") = Rep_BASE & "\LO"
WshEnv("SEMADMIN") = Rep_ABASE & "\LO"
WshEnv("PAC7ME") =  "NUL"
WshEnv("PAC7MV") =  Rep_TMP & "\WOJ.tmp"
WshEnv("PAC7RB") =  "NUL"
WshEnv("PAC7RY") =  "NUL"
Return = WshShell.Run("BVPACA15.exe" , 1, TRUE)
WshVolEnv("RC") = Return
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")
```

```
End If

End If 'If Rep_SAVE & "\PJ"

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

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

</script>
</job>
```

Chapter 3. Development Databases Management

Backup Procedures

Introduction

This procedure is used to perform different types of operations on the development Database data according to the input code entered on the '*' line.

PACS: User Inputs Common to Managers

Pos.	Len.	Value	Meaning
2	1	'*'	Line code
3	8	uuuuuuuu	User code
11	8	pppppppp	Password
22	4	nnnn	Session number (UXSR-reserved) blank = current session
26	1		Session type (UXSR-reserved)
		'T'	If selection of frozen session
		' '	If selection of current session
29	4	cccc	Procedure function code (1)
49	1		Locks extraction option (UXSR-reserved)
		' '	Locks extraction with user code = user code of '*' line
		'1'	No locks extraction
		'2'	Locks extraction with user code = source user code
67	1		(UXSR-reserved)
		'T'	If col 26 = ' ' selection of all the frozen sessions
		' '	IF col 26 = ' ' selection of the current session only

(1) The different values of function codes are:

- MLIB: library management
- SAVE: development Database backup
- SASN: sub-networks backup
- UXSR: sub-networks extraction

Database Management

MLIB: Introduction: The Database Management procedure has a two-fold purpose:

- Initialize the database in the form of a sequential file (or 2 files if the Dispatch option is used), called 'PC', which is then used as input to the Restoration (REST) procedure.
- Create or delete libraries in an existing database.

Execution conditions

The database must be closed to on-line access and use, unless the current execution is a simulation. The procedure must be followed by the REST procedure so that the new library structure is taken into account.

Abnormal executions

Once the problem has been solved, the procedure can be restarted as it is.

MLIB: Inputs - Processing - Results:

USER INPUT LINES There are two types of user input lines:

- Heading line (required) before all other input lines used to specify that a new VisualAge Pacbase Database is to be initialized or that an existing Database is to be modified.
- As many lines (optional) as there are Libraries to create, modify ou delete.

Pos.	Len.	Value	Meaning
2	1	'G'	Line code
3	1	' '	Modification of existing Database
		'I'	Initialization of new Database
4	1	' '	Actual update
		'S'	Simulation

Update simulation is used to obtain the state of the Database as it would appear if the requested modifications had actually been implemented.

It allows you to judge the impact of a change in the structure of the Database before actual execution. For large Databases, actual execution may use a lot of machine time.

The structure of the 'Library' lines is as follows:

Pos.	Lon.	Valeur	Meaning
1	1	'C'	Creation
		'M'	Modification
		'A'	Deletion
2	1	'*'	Line code
3	3	bbb	Code of the Library to update
6	3	ccc	Code of the upper level Library
9	1	' '	Not initialized Library
		'V'	Virtual Library
			(only for the creation)

NOTE :

Asterisks ("*") are not authorized in Library codes.

Update rules

Updates are executed line by line. No previous transaction sort is executed. The resulting Database must remain consistent during the update.

Deletion transactions:

A Library with dependent Libraries cannot be deleted. To delete an entire sub-network, begin by deleting the Libraries at the lowest hierarchical level and work upward to the highest level.

The upper Library code must not be entered on Library deletion lines. Only the code of the Library to be deleted may be specified.

The deletion of a Library causes this Library's entire contents to be deleted. Its contents are replaced by empty records, or 'gaps' (see the REST restoration procedure).

Creation transactions:

When a Library is created, it can only be linked to an already existing Library or to a Library that was previously created in the update job stream.

Therefore, always create the 'parent' Library before its 'child' Libraries. Both can be created by the same run of the procedure.

Once created, a Library has either a 'virtual' or 'not initialized' status.

- Virtual Libraries are created for future development projects. They are visible to the Administrator only. They cannot receive development specifications. You change their status to Not initialized in the Administrator workbench.
- Not initialized Libraries are visible to all users, but are not ready to receive development specifications. You change their status to Initialized in the Administrator workbench.

NOTE :

A VisualAge Pacbase Database cannot contain more than 300 Libraries.

Modification transactions

Generally, transactions modify links during the same run of the procedure.

When an error is detected on a line, a message is generated, and the update is interrupted because the resulting Database would otherwise be inconsistent. The line containing the error must be corrected and the job restarted, as the initial Database will not have been modified.

PRINTED REPORTS

In all cases, a report on the initial state of the Database and an update report are printed.

If no errors have been detected, a report on the Database is printed after the update.

RESULTS

If no errors are detected and if the update is 'real' (not simulated), the result is a sequential image of the updated Database (PC), which serves as input for Database reloading.

The execution report of a Database initialization does not include the Database name since it is later assigned via user input for the Restoration procedure.

WARNING

This procedure does not allow for the recovery of disk space when Libraries are deleted. Records are physically present in the Database as 'gaps'. It is the Reorganization (REOR) procedure that deletes these gaps so that disk space can be recovered.

NOTE :

This procedure increments the session number.

Database Backup

SAVE: Introduction: The Database Backup procedure (SAVE) performs a backup of the main files that make up the database. It produces sequential files with 'PC', 'PD' and 'PY' formats.

The backup is performed on the following files:

- Data file (AR),
- Index file (AN),
- The extension data file (AY).

An option allows for a database backup in two sequential files: one for the data ('AR' saved in 'PC' and 'AN' saved in 'PD'). Otherwise, the data and index files are saved in the 'PC' file only.

This option (Dispatch or No dispatch) is implemented in the database restoration procedure. For further details, see the REST procedure user input description.

Execution conditions

On-line access must be closed.

Abnormal executions

Refer to Chapter 'Overview', Subchapter 'Abnormal executions'.

The main cause of an abend is that the database remained open to on-line use while the procedure was being executed.

The procedure can be restarted as it is once the problem has been solved.

Archival and backup linking

If the backup procedure is preceded by a Journal archival (ARCH procedure), its execution may be conditioned by the return code of the PTU320 ARCH step:

- 0: No error detected
- 8: Invalid database

Simplified backup

Files may also be backed up via standard system utilities. In this case, run the SASY procedure to check the consistency of data and indexes (see Subchapter 'System Backup Complement').

Printed reports

The procedure prints:

- a report containing the number of records saved in each file and the session number),
- optional reports:
  - A statitical report with the number of records per library and per line type,
  - A report listings the database limits reached.

SAVE: Inputs - Processing - Results: The user may cancel the formatting and the output of statistical reports on the database, in order to speed up the execution of the backup procedure.

If a cancellation request is not made, all reports will be printed.

The structure of the line is as follows:

Pos.	Len.	Value	Meaning
2	2	'OR'	Line code
8	1		Statistical report by library of the database that has been backed up
		' '	Printing of statistics
		'N'	No printing of statistics
9	1		Report indicating the macro-structure call limitations in the database
		' '	Printing of limitations
		'N'	No printing of limitations

Output

The output of the backup procedure is the following:

- Either a single sequential file (PC), of variable length, containing the mirror of the two saved files,
- Or two sequential files, one of variable length containing the mirror of the data (PC), the other of fixed length containing the mirror of indices (its name depends on the platform).

If the database is no longer consistent after an abend during the last update, the backup procedure will not be executed.

If the database is inconsistent, the procedure sends back a return code.

Note:

This procedure increments the current session number.

Sub-Network Backup

SASN: Introduction: The Sub-Network Backup procedure (SASN) extracts one or several sub-networks from a database. The result is a consistent set of libraries which will make up a new database (formatted as a backup file to be used as input to the Restoration procedure).

Each extracted sub-network is identified by its lowest-level library; the utility automatically extracts all higher-level libraries pertaining to the sub-network.

Execution condition

The database must be closed to on-line use.

Abnormal executions

If an abend occurs, the procedure can be restarted as it is once the problem has been solved.

SASN: User Inputs: Batch procedure access authorization option: One '*' line with user code and password.

Pos.	Leng.	Value	Meaning
1	2	' '	
3	3	bbb	Code of lowest-level library of sub-network to be extracted. (All the upper-libraries of 'bbb' will be automatically extracted.)

Partial Sub-Network Extraction

UXSR: Introduction: The Partial Sub-Network Extraction procedure (UXSR) creates a VisualAge Pacbase sub-network from an existing database, by:

- Creating Libraries (MLIB equivalent)
- Merging Libraries
- Renaming Libraries

It is also possible to select:

- A frozen session (nT):

  This frozen session will become the current session in the new Database.

  No other frozen session will be selected.

  The image of this Database will be identical to the view which existed in the nT frozen session, but this time it will be in n+1 current session.
- The current session or all sessions (current included):

  Via an option, you can select all the sessions ('T' in position 67 of the * line), or only the current session (' ' in position 67 of the * line).

Examples:

- Creation of Libraries:

  C*CEN___AAA (1)

  C*APPCENBBB (2)

  (1) Creation of the CEN Library. AAA must not exist in the source Database.

  (2) Creation of the APP Library in the CEN Library. BBB must not exist in the source Database.

- Merging of Libraries in the same Library:

  C*CEN___CEN (1)

  C*APPCENAPP (2)

  C*APPCENBQQ (2)

  (1) Creation of the CEN Library with the contents of CEN.

  (2) Creation of the APP Library under the CEN Library with the contents of APP and BQQ.

  The definition of the APP Library in the new Database will be identical to that of APP in the source Database since APP comes first, before BQQ.
- Renaming of Library:

  C*CEN___AAA (1)

  (1) Creation of the CEN Library with the contents of APP.

Warning

No consistency checks are carried out; make sure you have entered valid user input lines.

It is not possible to copy an existing Library network and create new Libraries whose contents are identical to that of Libraries in the source network.

Execution conditions

On-line access must be closed.

This procedure processes data only. It must therefore be followed by the REOR, then REST procedures, in order for the new Database to be taken into account.

UXSR: User Inputs:

Batch procedure authorization access option: One '*' line with user code and password. There are two types of specific user input:

- Heading line (required) at the top of the input file that specifies a simulation or no of the database.
- As many lines (optional) as there are libraries to be created, modified or canceled.

Pos.	Len.	Value	Meaning
2	1	'G'	Line code
4	1	' '	Actual update
		'S'	Simulation

You must enter as many lines (optional) as Libraries to be extracted for update.

Pos.	Len.	Value	Meaning
1	1	'C'	Creation
2	1	'*'	Line code
3	3	bbb	Code of Library to be created
6	3	ccc	Code of higher Library if any

Pos.	Len.	Value	Meaning
9	3	ddd	Code of source Library required even when creating a new Library ; in this case enter any code not existing in the source Database.

Note: Do not use the character '*' in Library codes (incompatibility with the WorkStation).

PACS: Description of Steps

Formatting sequential images: PTU520

Code		Type	Label
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database index
PACGGR		Input	Administration Database data
PACGGU		Input	Administration Database users
PAC7AR		Input	Development Database data
PAC7AY		Input	Development Database extension data
PAC7AN		Input	Development Database index
PAC7MB		Input	Update transactions
PAC7PC		Output	Database sequential image (length=1,023)
PAC7PD		Output	If Dispatch option of backup: sequential image 2 of the database (length=1,023)
PAC7PY		Output	If Dispatch option of backup: sequential image 3 of the database (length=1,023)
PAC7RP		Output	Sequential image of data (length=153) (should be able to contain all data)
PAC7NA		Output	Sequential image of index (length=59) (should be able to contain all data)
PAC7NB		Output	Image od desorted index (length=59)
PAC7RY		Output	Sequential image of long data (length=1,019)
PAC7RQ		Output	Temporary storage (1 record, length=153)
PAC7EV		Report	User transactions list
PAC7EU		Report	State of library before and after
PAC7EW		Report	Backup report
PAC7DD		Report	Abnormal endings report

Return codes:

- 2: MLIB or SASN and no PTU530 execution error
- 4: MLIB and database simulation
- 8: Inconsistency of the database or no authorization for batch procedure

Formatting the sequential image: PTU530

Code		Type	Label
PAC7AR		Input	Development Database data

Code		Type	Label
PAC7RP		Input	Data sequential image
PAC7NA		Input	Index sequential image
PAC7NB		Input	unsorted index image
PAC7RY		Input	extension data sequential image
PAC7RQ		Input	Intermediary storage
PAC7PC		Output	Development Database sequential image
PAC7PD		Output	If backup Dispatch option: seqential image 2 of the network
PAC7PY		Output	If backup Dispatch option: sequential image 3 of the network

PACS: Execution Script

```
REM * -----------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -----------------------------------------------------
REM *                  - BACKUP OF THE DATABASE -
REM *
REM * -----------------------------------------------------
REM *
<job id=PACS>

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

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

<script language="VBScript">

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


Dim Ret520
Ret520 = 0

Call Msg_Log (Array("1022" , "PTU520"))
'-----------------------------------
WshEnv("PAC7MB") = Fic_Input
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7AY") = Rep_BASE & "\AY"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("SEMLOCK") = Rep_BASE & "\LO"
WshEnv("SEMADMIN") = Rep_ABASE & "\LO"
WshEnv("PAC7PC") = Rep_SAVE & "\PC.NEW"
WshEnv("PAC7PD") = Rep_SAVE & "\PCI.NEW"
WshEnv("PAC7PY") = Rep_SAVE & "\PCY.NEW"
WshEnv("PAC7NA") = Rep_TMP & "\WNA.tmp"
WshEnv("PAC7NB") = Rep_TMP & "\WNB.tmp"
WshEnv("PAC7RP") = Rep_TMP & "\WRP.tmp"
WshEnv("PAC7RQ") = Rep_TMP & "\WRQ.tmp"
WshEnv("PAC7RY") = Rep_TMP & "\WRY.tmp"
WshEnv("PAC7EU") = Rep_USR & "\PACSEU520.txt"
WshEnv("PAC7DD") = Rep_USR & "\PACSDD520.txt"
```

```
WshEnv("PAC7EW") = Rep_USR & "\PACSEW520.txt"
WshEnv("PAC7EV") = Rep_USR & "\PACSEV520.txt"
Return = WshShell.Run("BVPTU520.exe" , 1, TRUE)
Ret520= Return
If Return = 4 then Return = 0 end if

If Ret520 = 2 then
Call Msg_Log (Array("1022" , "PTU530"))
'-----------------------------------
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PAC7PC") = Rep_SAVE & "\PC.NEW"
WshEnv("PAC7PD") = Rep_SAVE & "\PCI.NEW"
WshEnv("PAC7PY") = Rep_SAVE & "\PCY.NEW"
WshEnv("PAC7NA") = Rep_TMP & "\WNA.tmp"
WshEnv("PAC7NB") = Rep_TMP & "\WNB.tmp"
WshEnv("PAC7RP") = Rep_TMP & "\WRP.tmp"
WshEnv("PAC7RQ") = Rep_TMP & "\WRQ.tmp"
WshEnv("PAC7RY") = Rep_TMP & "\WRY.tmp"
Return = WshShell.Run("BVPTU530.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTU530")
Else
If Return = 8 Then
Call Msg_Log (Array("1051"))
End If
Call Err_Cod(Return , 4 , "PTU520")
End If

If Ret520 <> 4 then
Call Msg_Log (Array("1022" , "BACKUP"))
'-----------------------------------
Call Turnover(Rep_SAVE & "\PC")
Call Turnover(Rep_SAVE & "\PCI")
Call Turnover(Rep_SAVE & "\PCY")
End If


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

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

</script>
</job>
```

UPDT: Freeze

UPDT: Introduction

The Database Update procedure (UPDT) is used to freeze the database and executes also a Batch update of the database.

The principle of sessions makes it possible to manage various versions of the same application. The administrator then freezes the Database, creating a snapshot of the current session.

The procedure allows access to all libraries which make up the database, according to the different user authorizations.

The update can be executed in on-line mode.

For further information about UPDT, refer to the 'Developer's Guide'.

UPDT: Inputs

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.

No other update should precede the 'X1HIST' line.

Pos.	Len.	Value	Meaning
2	6	'X1HIST'	Line code for a session freeze
8	50		Session Name
58	4		Session number
65	1		Session state
		' '	Visible and updated session
		'N'	Visible but not updated session
		'D'	Deleted session
66	15		Session Short Name

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

Code		Type	Label
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>
```

SASY: Database System Backup Complement

SASY: Introduction

This Database Backup procedure, called System, is used to save the Database using any utility of the Operating System, while at the same time creating a checkpoint, through the incrementation of the session number.

The following files are to be backed up:

- Data file (AR),
- Index file (AN).

Execution conditions

The Data (AR) and Index (AN) files must have been backed up.

The transaction Journal file (AJ) must have been archived via the ARCH procedure.

The database must be closed to on-line use in order to maintain its consistency during the backup.

Abnormal endings

The main cause of an abend is that the database remained open to on-line use while the procedure was being executed.

The procedure can be restarted as it is once the problem has been solved.

User input

No user input is necessary when requesting the execution of the SASY procedure.

Result

This procedure increments the current session number.

If the database is in an inconsistent state due to an abend in the last update, the SASY procedure is not executed and the backup executed by the on-site Operating System utility is not valid.

SASY: Description of Steps

Session number icrementation: PTU502

Code		Type	Label
PAC7AR		Input/Output	Development Database data
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database index
PACGGR		Input	Administration Database data
PACGGU		Input	Administration Database users
PAC7MB		Input	User transaction
PAC7GZ		Report	Report
PAC7DD		Report	Batch procedure authorization option

Code		Type	Label
PAC7DS		Report	Database validity report

SASY: Execution Script

```
REM * -----------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -----------------------------------------------------
REM *     - COMPLEMENT OF 'SYSTEM' BACKUP OF THE DATABASE-
REM *
REM * -----------------------------------------------------
REM *
REM *  THE DATABASE SYSTEM BACKUP COMPLEMENT PROCEDURE
REM *  (SASY) ALLOWS YOU TO SAVE THE DATABASE USING ANY
REM *  UTILITY OF THE OPERATING SYSTEM, WHILE AT THE
REM *  SAME TIME CREATING A CHECKPOINT, THROUGH THE
REM *  INCREMENTATION OF THE SESSION NUMBER.
REM *  THE FOLLOWING FILES ARE TO BE BACKED UP:
REM *    . DATA FILE (AR),
REM *    . INDEX FILE (AN).
REM *
REM * -----------------------------------------------------
REM *
<job id=SASY>

<script language="VBScript">
Dim MyProc
MyProc = "SASY"
</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", "PTU502"))
'-------------------------------------
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("PAC7DD") = Rep_USR & "\SASYDD502.txt"
WshEnv("PAC7DS") = Rep_USR & "\SASYDS502.txt"
WshEnv("PAC7GZ") = Rep_USR & "\SASYGZ502.txt"
Return = WshShell.Run("BVPTU502.EXE" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTU502")


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


</script>
</job>
```

REST: Database Restoration

REST: Introduction

The Database Restoration procedure (REST) re-creates a database that can be manipulated on-line, using the sequential image produced by the Backup, the Database Management (PACS), the Reorganization (REOR) and Storage Optimization of Multi-volume Data (STOP) procedures.

It also allows the retrieval of archived transactions after this sequential image has been produced.

Execution conditions

The database must be closed to on-line processing.

recommended

The REST procedure physically and logically reinitializes the Journal file, which must have been saved previously by the ARCH procedure.

Abnormal executions

Refer to chapter 'Overview', subchapter 'Abnormal Executions'.

If an abend occurs, the procedure can be restarted as it is once the problem has been solved.

REST: Inputs - Processes - Results

Batch procedure access authorization: one '*' line with user code and password.

The structure of the specific input is described in the chart below.

Pos.	Len.	Value	Meaning
2	1	Y	Line code
3	5	nnnnn	Number of unused gaps
8	2	pp	Number of unused gaps as a percentage
10	2		Language code (EN, FR, ...)
12	1	0 1 ' '	No suppression of journal Suppression of journal (no journali zation of update transactions) Previous value
14	3	REC	If archived transactions recovered
17	4	XXXX	4-character Database code chosen by the Database Manager (displayed in the top-right corner of VA Pac screens) Database code is required
21	4	nnnn	Maximum access number: on-line search (lists) (default value: 300)
25	1	U	Implicit update (default option)
		N	Explicit update
36	12		PFkeys assigned functions (2).
79	1		Dispatch option of Backup:

Pos.	Len.	Value	Meaning
		'D'	Dispatch: sequential backup of the database in two separate files.
		'N'	No Dispatch: standard backup of the database in one PC file.
		' '	Same value as previous restoration

(2): PFKeys assignment:

12-position table, with each position referring to a standard function.

To modify the PFkey assigned to a function, the value of the new PFkey coded in base 36 is entered in the corresponding position in the table.

For example, to assign function 1 to PFKey 17, enter code 'H' in position 1 of the table.

No validation procedure is executed by the system. The PFkey assignment may be viewed on the corresponding sub-menu.

Notes:

Where there is no input, the Database characteristics remain unchanged.

Any field left blank will retrieve the current options.

- The limit of on-line accesses to the Journal depends on the number specified as input of the restoration procedure.

  If you do not want the update transactions of the database to be saved in the Journal file, you can turn the 'journalization' off by setting this parameter to '1'. In this case, it is not possible to restore the database using the recovery of archived transactions ('REC' entered on the input parameter card). It is therefore highly recommended to set this parameter to 0 (which is the default option), in order to avoid restoration problems.

  In case of error, invalid parameters are ignored, and the system ensures restoration using the parameter values stored in the sequential image of the database.

Simplified restoration

If the backup was performed via a system utility followed by the SASY procedure, restoration via a utility must be followed by the RESY procedure, which ensures the consistency between files.

Output reports

This procedure prints a report listing the requested options, associated errors, the number of records restored in the database for each file and the options stored in the new database.

General results

Once the procedure has been executed, the database is ready to be used in batch or on-line mode.

Note:

Once this procedure is executed, the current session number is the same as the session number of the sequential image, or of the most recent transaction, if you've requested archived transaction retrieval.

REST: Description of Steps

User input recognition: PTU010

Code		Type	Label
CARD		Input	User parameters
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database index
PACGGR		Input	Administration Database data
PACGGU		Input	Administration Database users
PAC7PC		Input	Sequential image of the development Database
PAC7PD		Input	If backup option Dispatch: sequential image 2 of the database
PAC7PY		Input	If Dispatch option of the backup: sequential image 3 of the database
PAC7MB		Output	Parameter
PAC7DD		Report	Batch procedures authorization option

Return code:

- 8: No batch procedure authorization

Validation of journal contents: PTU380

This step is executed only if the Journal file exists.

Code		Type	Label
PAC7MB		Input	User transactions
PAC7AE		Input	Error messages file
PAC7AJ		Input	Journal file
PAC7EU		Report	(only if the journal was not backed-up)

. Return code:

- 0: The journal file was archived
- 8: The journal file was not archived (no REST step is executed)

Restoration of the database: PTU400

This step is executed only if the Journal file has been archived.

Code		Type	Label
PAC7AE		Input	Error messages
PAC7MB		Input	User transaction
PAC7PC		Input	Sequential image of the development Database

Code		Type	Label
PAC7PD		Input	If backup option Dispatch: sequential image 2 of the database
PAC7PY		Input	If backup option Dispatch: sequential image 3 of the database
PAC7AR		Input	Development Database data
PAC7AN		Output	Development Database index
PAC7AY		Output	Development Database extension data
PAC7AJ		Output	Development Database Journal
PAC7PS		Output	Working file (2 records, length=144)
PAC7EU		Report	Backup report
PAC7DD		Report	Batch procedures authorization option

Database availability - transaction retrieval: PTU420

This step is executed if the Journal file has been archived. It updates the first record of the Data file.

This step is REQUIRED to obtain a consistent database.

Code		Type	Label
PAC7AE		Input	Error messages
PAC7MB		Input	User transactions
PAC7AR		Input/Output	Administration Database data
PAC7JO		Input	Journal to apply
PAC7PS		Input	Working file
PAC7OJ		Output	Update transactions (length=170)
PAC7EU		Report	Retrieval report

Return codes:

- 0: There are transactions to retrieve.
- 4: No transactions to retrieve OR erroneous user input.

In case of abend, the update cannot be performed.

Updating the development Database: PACA15

Code		Type	Label
PAC7AR		Output	Development Database data
PAC7AN		Output	Development Database index
PAC7AY		Output	Development Database extension
PAC7AJ		Output	Development Database Journal
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database index
PACGGR		Input	Administration Database data
PACGGY		Input	Administration Database extension

Code		Type	Label
PACGGU		Input	Administration Database users
PAC7DC		Input	DSMS elements file of the development Database
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	Retrieval of erroneous transactions (length=132)

The list of transactions belonging to a user is preceded by a banner specifying the user code.

REST: Execution Script

```
REM * -----------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -----------------------------------------------------
REM *         - RELOADING RESTORATION OF THE DATABASE -
REM *
REM * -----------------------------------------------------
REM *
REM * INPUT
REM * COL  2     : 'Y'
REM * COL  3-7   : NUMBER OF GAPS IN ABSOLUTE VALUE
REM * COL  8-9   : NUMBER OF GAPS IN PERCENTAGE   ( / BASE )
REM * COL 10-11  : INITIAL LANGUAGE CODE (FR, EN, ..)
REM * COL 12     : '1' INHIBITION OF TRANSACTION LOG
REM * COL 14-16  :'REC' FOR RECOVERY OF ARCHIVED TRANSACTIONS
REM * COL 17-20  : 4 CHARACTERS TO BE DISPLAYED
REM *              ON ALL SCREEN OF THE PRODUCT
REM * COL 21-24  : 'NNNN' MAXIMUM NUMBER OF SEARCH ACCESSES
REM *              TO THE DATABASE(LISTS)-(DEFAULT VALUE:300)
REM * COL 25     : 'U' (DEFAULT VALUE) : IMPLICIT UPDATE
REM *            : 'N' EXPLICIT UPDATE
REM * COL 26-29  : CKECKPOINT FREQUENCY
REM * COL 36-47  : PF-KEYS SIGNIFICATIONS
REM * COL 79     : BACKUP FILES DISPATCH
REM *            : 'N' (DEFAULT VALUE) : NO DISPATCH (1 FILE)
REM *            : 'D' : DISPATCH (3 FILES)
REM *
REM * IN THE ABSENCE OF INPUT, THE RELOAD DOES NOT MODIFY THE
REM * NUMBER OF EXISTING GAPS, AND OTHER DATA IS UNCHANGED.
REM *
REM * IF THE JOURNAL FILE OF TRANSACTIONS ON DISK (AJ) IS NOT
REM * REINITIALIZED, THE RESTORE CHAIN IS NOT EXECUTED.
REM * IT IS THEREFORE NECESSARY TO EXECUTE THE ARCH PROCEDURE
REM * FIRST.
REM * -----------------------------------------------------
<job id=REST>

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

<script language="VBScript">
```

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


Dim Ret420
Ret420 = 0

Call Msg_Log (Array("1022" , "PTU010"))
'-----------------------------------
WshEnv("CARTE") = Fic_Input
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7DD") = Rep_USR & "\RESTDD010.txt"
WshEnv("PAC7MB") = Rep_TMP  & "\WMB.tmp"
WshEnv("PAC7PC") = Rep_SAVE & "\PC"
WshEnv("PAC7PD") = Rep_SAVE & "\PCI"
WshEnv("PAC7PY") = Rep_SAVE & "\PCY"
Return = WshShell.Run("BVPTU010.exe" , 1, TRUE)
WshVolEnv("RC") = Return

If Return = 8 Then
Call Msg_Log (Array("1027"))
End If
Call Err_Cod(Return , 0 , "PTU010")

If  FSO.FileExists(Rep_JOURNAL & "\AJ") Then
Call Msg_Log (Array("1022", "PTU380"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AJ") =  Rep_JOURNAL & "\AJ"
WshEnv("PAC7EU") =  Rep_USR & "\RESTEU380.txt"
WshEnv("PAC7MB") = Rep_TMP  & "\WMB.tmp"
Return = WshShell.Run("BVPTU380.EXE" , 1, TRUE)
WshVolEnv("RC") = Return
If Return = 8 Then
Call Msg_Log (Array("1058"))
End If
Call Err_Cod(Return , 0 , "PTU380")
End If

Call Msg_Log (Array("1022" , "PTU400"))
'-----------------------------------
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("PAC7DD") =  Rep_USR & "\RESTDD400.txt"
WshEnv("PAC7EU") =  Rep_USR & "\RESTEU400.txt"
WshEnv("PAC7PC") = Rep_SAVE & "\PC"
WshEnv("PAC7PD") = Rep_SAVE & "\PCI"
WshEnv("PAC7PS") =  Rep_TMP & "\WPS.tmp"
WshEnv("PAC7PY") = Rep_SAVE & "\PCY"
Return = WshShell.Run("BVPTU400.exe" , 1, TRUE)
WshVolEnv("RC") = Return
Call Err_Cod(Return , 0 , "PTU400")

If  FSO.FileExists( Rep_SAVE & "\PJ"  ) Then
Call Msg_Log (Array("1022" , "PTU420"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AR") =  Rep_BASE & "\AR"
WshEnv("PAC7JO") =  Rep_SAVE & "\PJ"
WshEnv("PAC7MB") =  Rep_TMP & "\WMB.tmp"
```

```
WshEnv("PAC7OJ") =  Rep_TMP & "\WOJ.tmp"
WshEnv("PAC7PS") =  Rep_TMP & "\WPS.tmp"
WshEnv("PAC7EU") =  Rep_USR & "\RESTEU420.txt"
Return = WshShell.Run("BVPTU420.EXE" , 1, TRUE)
Ret420 = Return

If Return = 4 Then
'No transaction to be retrieved : Normal End
'-----------------------------------------
Call Msg_Log (Array("1059"))
Call Msg_Log (Array("1024"))
Return = 0
WshVolEnv("RC") = Return

Else
WshVolEnv("RC") = Return
Call Err_Cod(Return , 0 , "PTU420")
End If

If Ret420 = 0 then
'Update
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("PACGGY") = Rep_ABASE & "\AY"
WshEnv("PAC7DC") =  Rep_BASE & "\DC"
WshEnv("PAC7IE") =  Rep_USR & "\RESTIEA15.txt"
WshEnv("PAC7IF") =  Rep_USR & "\RESTIFA15.txt"
WshEnv("SEMLOCK") = Rep_BASE & "\LO"
WshEnv("SEMADMIN") = Rep_ABASE & "\LO"
WshEnv("PAC7ME") =  "NUL"
WshEnv("PAC7MV") =  Rep_TMP & "\WOJ.tmp"
WshEnv("PAC7RB") =  "NUL"
WshEnv("PAC7RY") =  "NUL"
Return = WshShell.Run("BVPACA15.exe" , 1, TRUE)
WshVolEnv("RC") = Return
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")
End If

End If 'If Rep_SAVE & "\PJ"

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

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

</script>
</job>
```

RESY: Database System Restoration Complement

RESY: Introduction

This procedure restores a Database that can be handled in on-line mode, from a System backup obtained through a utility followed by the SASY procedure.

It is executed after a System restoration utility to complete the restoration of the Data (AR) and Index (AN) files, and reinitializes the Journal (AJ) file.

Through the RESY procedure, the archived transactions can be recovered if 'REC' is entered on the input parameter line.

If the Journal file is not reinitialized, it must be archived prior to the System utility restoration and RESY procedures.

Execution conditions

This procedure can be executed only after restoration of the AN and AR files by the on-site system utility.

On-line access must be closed.

Abnormal executions

If an abend occurs, the procedure can be restarted as it is once the problem has been solved.

Printed results

The RESY procedure prints a report listing the requested options and related errors, the number of records reloaded in the database per file and the options memorized in the new database.

General results

Once the RESY procedure has been executed, the database can be used in both batch or on-line modes.

Note: After the procedure execution, the current session number is the session number of the restored image, or of the most recent transaction if the retrieval of archived transactions has been requested.

RESY: Inputs - Processing - Results

A line '*' with user code and password.

The input has the following structure:

Pos.	Lon.	Valeur	Meaning
2	1	'Y'	Line code
10	1		Language code (FR, EN...)
12		0 '1' ' '	No journal suppression Suppression of journal (update transactions are not journalized) Retrieval of the last value
14	3	REC	if archived transactions are recov'd.

Pos.	Lon.	Valeur	Meaning
17	4	XXXX	4-character Database code chosen by the Database Manager (displayed in the top-right corner of all screens)
21	4	'nnnn'	Maximum access number: on-line search (lists) (default value: 300)
25	1	U	Implicit update (default option)
		N	Explicit update
26	4	'nnnn'	Checkpoint frequency rate (IMS, UNISYS, GCOS7, and GCOS8 only) if REC in col. 13 (default: nnnn=0000)
36	12		PFkeys assigned functions (1)
79	1		Dispatch option of backup:
		'D'	Dispatch Sequential backup of the database on two separate files.
		'N'	No Dispatch Standard backup on a single PC file.
		' '	Same as previous execution.

PFKeys assignment:

12-position table: each position corresponds to a standard function. To modify the PFkey assigned to a function, the value of the new PFkey coded in base 36 is entered in the corresponding position in the table.

For example, to assign function 1 to PFkey 17, code 'H' in position 1 of the table.

No validation procedure is executed by the system. The PFkey assignment may be viewed on the corresponding sub-menu.

Note:

Any field left blank defaults to the current option selection.

If you do not want the update transactions of the database to be saved on the Journal file, you can turn "journalization" off by setting this parameter to '1'. In this case, it is not possible to restore the database using the recovery of the archived transactions (REC parameter in the user input).

It is thus highly recommended that you set this parameter to '0' or leave it blank (default option), in order to avoid restoration problems.

In case of error, invalid parameters are ignored, and the system ensures restoration using the parameter values stored in the sequential image of the database.

RESY: Description of Steps

User input recognition: PTU004

Code		Type	Label
CARD		Input	Parameter
PAC7MB		Output	Parameter
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database index

Code		Type	Label
PACGGR		Input	Administration Database data
PACGGU		Input	Administration Database users
PAC7AR		Input	Development Database data
PAC7DD		Report	Batch procedures authorization option

Return code:

- 8: No authorization for batch procedure

Validation of journal contents: PTU380

This step is executed only if the Journal file exists.

Code		Type	Label
PAC7MB		Input	User transactions
PAC7AE		Input	Error messages file
PAC7AJ		Input	Journal file
PAC7EU		Report	(only if the journal was not backed-up)

. Return code:

- 0: The journal file was archived
- 8: The journal file was not archived (no REST step is executed)

Database positioning: PTU402

This step is executed only if the Journal file has been archived.

Code		Type	Label
PAC7AR		Output	Data file
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database index
PACGGR		Input	Administration Database data
PACGGU PACGGY		Input Input	Administration Database users
PAC7MB		Input	User transactions
PAC7PS		Output	Working file (2 records, length=144)
PAC7GZ		Report	Backup report

Database availability - transaction retrieval: PTU420

This step is executed if the Journal file has been archived. It updates the first record of the Data file.

This step is REQUIRED to obtain a consistent database.

Code		Type	Label
PAC7AE		Input	Error messages

Code		Type	Label
PAC7MB		Input	User transactions
PAC7AR		Input/Output	Administration Database data
PAC7JO		Input	Journal to apply
PAC7PS		Input	Working file
PAC7OJ		Output	Update transactions (length=170)
PAC7EU		Report	Retrieval report

Return codes:

- 0: There are transactions to retrieve.
- 4: No transactions to retrieve OR erroneous user input.

In case of abend, the update cannot be performed.

Updating the development Database: PACA15

Code		Type	Label
PAC7AR		Output	Development Database data
PAC7AN		Output	Development Database index
PAC7AY		Output	Development Database extension
PAC7AJ		Output	Development Database Journal
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database index
PACGGR		Input	Administration Database data
PACGGY		Input	Administration Database extension
PACGGU		Input	Administration Database users
PAC7DC		Input	DSMS elements file of the development Database
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	Retrieval of erroneous transactions (length=132)

The list of transactions belonging to a user is preceded by a banner specifying the user code.

RESY: Execution Script

```
REM * -----------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -----------------------------------------------------
REM *        - 'SYSTEM' RELOADING RESTORATION COMPLEMENT -
REM *
REM * -----------------------------------------------------
REM *
REM *
```

```
REM * INPUT
REM * COL  2     : 'Y'
REM * COL 10-11  : INITIAL LANGUAGE CODE (FR, EN, ..)
REM * COL 12     : '1' INHIBITION OF TRANSACTION LOG
REM * COL 14-16  :'REC' FOR RECOVERY OF ARCHIVED TRANSACTIONS
REM * COL 17-20  : 4 CHARACTERS TO BE DISPLAYED ON ALL
REM *                SCREEN OF THE PRODUCT
REM * COL 21-24  : 'NNNN' MAXIMUM NUMBER OF SEARCH ACCESSES
REM *              TO THE DATABASE(LISTS)-(DEFAULT VALUE:300)
REM * COL 25     : 'U' (DEFAULT VALUE) : IMPLICIT UPDATE
REM *            : 'N' EXPLICIT UPDATE
REM * COL 26-29  : CKECKPOINT FREQUENCY
REM * COL 36-47  : PF-KEYS SIGNIFICATIONS
REM * COL 79     : BACKUP FILES DISPATCH
REM *            : 'N' (DEFAULT VALUE) : NO DISPATCH (1 FILE)
REM *            : 'D' : DISPATCH (2 FILES)
REM *
REM * IF THE JOURNAL FILE OF TRANSACTIONS ON DISK (AJ) IS NOT
REM * REINITIALIZED, THE RESTORE CHAIN IS NOT EXECUTED.
REM * IT IS THEREFORE NECESSARY TO EXECUTE THE ARCH
REM * PROCEDURE FIRST.
REM *
REM * -----------------------------------------------------
REM *
<job id=RESY>

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

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

<script language="VBScript">

If c_error = 1 then
  WshVolEnv("RC") = 1
  Wscript.Quit (1)
End If

Call Msg_Log (Array("1022" , "PTU004"))
'-----------------------------------
WshEnv("CARTE") = 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("PAC7DD") = Rep_USR & "\RESYDD004.txt"
WshEnv("PAC7MB") = Rep_TMP  & "\WMB.tmp"
Return = WshShell.Run("BVPTU004.exe" , 1, TRUE)
WshVolEnv("RC") = Return
If Return = 8 Then
Call Msg_Log (Array("1027"))
End If
Call Err_Cod(Return , 0 , "PTU004")


If  FSO.FileExists(Rep_JOURNAL & "\AJ") Then
Call Msg_Log (Array("1022", "PTU380"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AJ") =  Rep_JOURNAL & "\AJ"
WshEnv("PAC7EU") =  Rep_USR & "\RESYEU380.txt"
WshEnv("PAC7MB") = Rep_TMP  & "\WMB.tmp"
Return = WshShell.Run("BVPTU380.EXE" , 1, TRUE)
WshVolEnv("RC") = Return
```

```
If Return = 8 Then
Call Msg_Log (Array("1058"))
End If
Call Err_Cod(Return , 0 , "PTU380")

Call Msg_Log (Array("1022" , "PTU402"))
'-----------------------------------
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("PACGGY") = Rep_ABASE & "\AY"
WshEnv("PAC7GZ") = Rep_USR & "\RESYGZ402.txt"
WshEnv("PAC7MB") = Rep_TMP & "\WMB.tmp"
WshEnv("PAC7PS") = Rep_TMP & "\WPS.tmp"
Return = WshShell.Run("BVPTU402.exe" , 1, TRUE)
WshVolEnv("RC") = Return
Call Err_Cod(Return , 0 , "PTU402")

If  FSO.FileExists( Rep_SAVE & "\PJ"  ) Then
Call Msg_Log (Array("1022" , "PTU420"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AR") =  Rep_BASE & "\AR"
WshEnv("PAC7JO") =  Rep_SAVE & "\PJ"
WshEnv("PAC7MB") =  Rep_TMP & "\WMB.tmp"
WshEnv("PAC7OJ") =  Rep_TMP & "\WOJ.tmp"
WshEnv("PAC7PS") =  Rep_TMP & "\WPS.tmp"
WshEnv("PAC7EU") =  Rep_USR & "\RESTEU420.txt"
Return = WshShell.Run("BVPTU420.EXE" , 1, TRUE)
Ret420 = Return

If Return = 4 Then
'No transaction to be retrieved : Normal End
'-----------------------------------------
Call Msg_Log (Array("1059"))
Call Msg_Log (Array("1024"))
Return = 0
WshVolEnv("RC") = Return

Else
WshVolEnv("RC") = Return
Call Err_Cod(Return , 0 , "PTU420")
End If

If Ret420 = 0 then
'Update
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("PACGGY") = Rep_ABASE & "\AY"
WshEnv("PAC7DC") =  Rep_BASE & "\DC"
WshEnv("PAC7IE") =  Rep_USR & "\RESTIEA15.txt"
WshEnv("PAC7IF") =  Rep_USR & "\RESTIFA15.txt"
WshEnv("SEMLOCK") = Rep_BASE & "\LO"
WshEnv("SEMADMIN") = Rep_ABASE & "\LO"
WshEnv("PAC7ME") =  "NUL"
WshEnv("PAC7MV") =  Rep_TMP & "\WOJ.tmp"
WshEnv("PAC7RB") =  "NUL"
```

```
WshEnv("PAC7RY") =  "NUL"
Return = WshShell.Run("BVPACA15.exe" , 1, TRUE)
WshVolEnv("RC") = Return
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")
End If

End If 'If Rep_SAVE & "\PJ"

End If
'/ ? existing AJ

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

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

</script>
</job>
```

ARCH: Journal Archival

ARCH: Introduction

This procedure backs up the Journal file as a sequential file, and re-initializes it both logically and physically.

Archived transactions do not override those transactions that were previously archived, but rather are added to them.

The archived-transaction file may be purged. Purged transactions may then be saved in another file (PQ).

Previously archived transactions can be purged, if requested. (However, non-archived journal transactions cannot be purged.)

EXECUTION CONDITIONS

On-line access must be closed.

ABNORMAL EXECUTIONS

If an abend occurs before the step that creates the Journal file, the procedure can be restarted as it is once the problem has been solved.

Otherwise, the procedure must be restarted after a modification of the user input in order to specify a re-initialization request without a backup of the Journal file, since it has already been backed up.

ARCH: Inputs - Processing - Results

Batch procedure access authorization option: one '*' line with user code and password.

This procedure includes specific optional input for:

- Purging previously archived transactions that are considered obsolete.
- Signalling the absence of previously archived transactions during input.
- Signalling the unavailability of the Data file (AR) during input.
- Requesting the re-initialization of the transaction file only.

The structure of this input is as follows:

POS.	LEN.	VALUE	MEANING
2	1	'S'	Line code
3	4	nnnn	Session number
7	8	ccyymmdd	OR date up to which the user requests deactivation
15	1	'I'	Absence of previously archived transactions
16	1	'D'	Data file unavailable
17	1	'J'	Re-initialization without backup, the transactions already archived are NOT retrieved on output.

The session number and the date are independent of each other. They are ignored if it is indicated that there are no input transactions (refer to Section Recommendations).

The unavailability of the Data file is to be indicated only when this file has been physically deleted. (See Section Recommendations below.)

A request to re-initialize without archiving is necessary when the Journal file is physically deleted.

Note:

In this case, the transactions which were already archived are not copied on to the transaction output file. (If this input file is automatically catalogued by the operating system, the transactions already archived may be lost unless the file is uncatalogued).

In an error occurs on one of the options, a message is printed and the archive is generated using the default options.

Recommentations

If there is no user input, this procedure can only be executed if the Database is in a consistent state, and if the archived transaction file is correctly formatted.

When the Database needs to be restored after an abend or a system failure, some information in the Specifications Dictionary is sometimes lost, thus preventing the execution of the backup and restauration procedures.

In this case, AND IN THIS CASE ONLY, columns 15 to 17 of the user input are to be used as follows:

- If the Data file (AR) is lost or has been flagged as 'inconsistent', a 'D' in column 16 means that the backup procedure will not take the Data file (AR) into account. However, the restoration procedure must be executed afterward, since under these conditions, the backup procedure leaves the database in an inconsistent state.
- If the Journal file (AJ) is lost or destroyed, a 'J' must be entered in column 17. As a result, the backup procedure formats an empty Journal file. The restoration procedure may then be executed (not compulsory). In this case, the content of the journal file (AJ) is lost.
- If the transactions Back-up file is lost or destroyed, an 'I' must be entered in column 15. As a result, the ARCH procedure reformats a new sequential backup file of (archived) transactions.

If one of these columns is accidentally set, and if the ARCH procedure is executed while the Database is in a consistent state, the consequences are:

- 'I' in col. 15: Previously archived transactions are lost. All transactions can be recovered by concatenating (-1) and (0) to obtain (+1).
- 'D' in col. 16: The ARCH procedure must be re-executed BEFORE any update. If an update is subsequently performed, the Database will be lost, and will have to be restored completely.
- 'J' in col. 17: The contents of the Journal file are definitely lost. The output Journal file, (+1 version in the case of generation data files), is created empty.

Printed output

This procedure prints a report stating the number of archived transactions and, if applicable, the number of records that have been 'purged'.

Results

Once this procedure is executed, a sequential file containing all archived transactions is obtained.

The Journal file which displays transactions on-line is re-initialized.

It is also possible to store on another file all transactions that have been purged.

Note:

This procedure does not increment the session number.

ARCH: Description of Steps

PARTICULAR CASE OF THE FIRST DATABASE ARCHIVING

In order for the first database archival to run correctly, the PJ file, containing archived transactions used as input of the procedure, is created as an empty file, and stored in the ÈSAVE directory during installation.

ARCHIVED-TRANSACTION PURGE

When a purge of archives is requested in the transactions files, two situations are possible:

1. The user does not wish to keep the purged archives in the PJ file: the file with PAC7PQ as internal name must be assigned to 'NUL'.

This is done as a default in the procedure command file.

2. The user wishes to keep purged archives in the PJ file: the file with PAC7PQ as internal name must be assigned.

   For instance:

   WshEnv(″PAC7PQ) = Rep_SAVE & ″\PQ″

Archival of journal file: PTU300

This step:

- writes obsolete transactions to be purged on to a special file, if the purge is requested in user input.
- positions a flag in the Data file indicating the journal archive.
- updates the file of archived transactions.

Code		Type	Label
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database index
PACGGR		Input	Administration Database data
PACGGU		Input	Administration Database users
PAC7JP		Input	Previously archived transactions
PAC7AJ		Input	Development Database journal to re-initialize
PAC7MB		Input	User transaction
PAC7BM		Output	User transaction
PAC7AR		Input/Output	Development Database data
PAC7PJ		Output	Archived update transactions
PAC7PQ		Output	Deactivated transactions(length=170) modify the file name in order to keep the transactions
PAC7EU		Report	Output report
PAC7DD		Report	Batch procedures authorization option

. Return codes:

- 0: No error detected on the files
- 4: Erroneous record of the journal file
- 8: No batch procedure access authorization OR invalid Database; restart the procedure with 'D' in column 16 of the user input (MBARCH.bvp).
- 12: Input-output error on a file

```
Re-initialization of the journal file:  PTU320
This step executes the following:
```

- Creates the first record in the Journal file,
- Re-initializes the Data file flag with the Journal file's address.

Code		Type	Label
PAC7BM		Input	User transaction
PAC7AR		Input/Output	Data file
PAC7AE		Input	Error messages file
PAC7AJ		Output	Journal file to re-initialize

Code		Type	Label
PAC7EU		Report	Re-initialization report

.Return codes: 0: No error detected, 8: The database is not available.

If the ARCH and PACS SAVE option procedures are grouped into one job, this return code can be tested in order to condition the execution of the SAVE procedure.

ARCH: Execution Script

```
REM * -----------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -----------------------------------------------------
REM *             - ARCHIVAL OF THE JOURNAL -
REM *
REM * -----------------------------------------------------
REM *
REM * INPUT       : COMMAND FOR DEACTIVATION OF ARCHIVED
REM *               TRANSACTION
REM * COL 2       : 'S'
REM * COL 3 TO  6 : SESSION NUMBER
REM * COL 7 TO 14 : DATE (CCYYMMDD)
REM * COL 15      : ' ' PRESENCE OF ARCHIVED TRANSACTION FILE
REM *             : 'I' ABSENCE OF ARCHIVED TRANSACTION FILE
REM * COL 16      : ' ' PRESENCE OF DATA FILE (AR)
REM *             : 'D' ABSENCE OF DATA FILE (AR)
REM * COL 17      : ' ' ARCHIVAL AND REINITIALIZATION
REM *             : 'J' REINITIALIZATION WITHOUT ARCHIVAL
REM *
REM * IN THE ABSENCE OF INPUT (OR ERROR ON A COMMAND PARAM.)
REM * NO DEACTIVATION WILL TAKE PLACE, HOWEVER ARCHIVAL AND
REM * REINITIALIZATION WILL BE EXECUTED NORMALLY.
REM *
REM * TRANSACTIONS WHOSE SESSION (DATE) IS PRIOR OR EQUAL TO
REM * THE SESSION (DATE) INDICATED ARE NOT KEPT. THEY ARE
REM * RECOVERED IN THE FILE OF DEACTIVATED TRANSACTION.
REM *
REM * -----------------------------------------------------
<job id=ARCH>

<script language="VBScript">
MyProc = "ARCH"
</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" , "PTU300"))
'-------------------------------------
WshEnv("PAC7MB") = Fic_Input
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PAC7AJ") = Rep_JOURNAL & "\AJ"
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7JP") = Rep_SAVE & "\PJ"
WshEnv("PAC7PJ") = Rep_SAVE & "\PJ.new"
WshEnv("PAC7PQ") = "NUL"
```

```
WshEnv("PAC7BM") = Rep_TMP & "\WBM.tmp"
WshEnv("PAC7EU") = Rep_USR & "\ARCHEU300.txt"
WshEnv("PAC7DD") = Rep_USR & "\ARCHDD300.txt"
Return = WshShell.Run("BVPTU300.exe" , 1, TRUE)
If Return = 8 Then
Call Msg_Log (Array("1032"))
End If
If Return = 12 Then
Call Msg_Log (Array("1026"))
End If
Call Err_Cod(Return , 0 , "PTU300")



Call Msg_Log (Array("1022" , "PTU320"))
'-----------------------------------
WshEnv("PAC7EU") = Rep_USR & "\ARCHEU320.txt"
Return = WshShell.Run("BVPTU320.EXE" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTU320")

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

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

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

</script>
</job>
```

REOR: Reorganization

REOR: Introduction

The Database Reorganization procedure optimizes Database accesses by accounting for each deletion, and sorting the data again according to the most frequent access order.

It uses one or two backup files of the development Database (if the Dispatch option is used), to rebuild one (or 2) sequential image(s). This resulting image file must then be restored via the REST procedure described above.

The operating principle of this procedure is to rebuild the different indexes associated with all data using the 'image' of each Data Element. It makes the best of the system performance features since it separates historical (frozen) sessions from the current session and sorts the data in the order of the most frequent access. This makes it possible to achieve a significant reduction of the number of indexes and data items.

The REOR procedure may be used in two cases:

- When part of the data was deleted because of a malfunction or system failure, and no other procedure can be used (in particular, deletion of the AN Index file),
- When the database is to be purged of the following:
  - Obsolete libraries and/or sessions;
  - Entities not used in the database;

When a library is deleted, this procedure produces the same results as the Database Management (MLIB) procedure, except that it additionally deletes 'gaps'.

This procedure should be executed only on an exceptional basis, because of the special conditions concerning its use and its lengthy execution time.

Deletions taken into account by the reorganization may have been made logically by the Database update, or generated by one or several utilities. For example:

- Deletion of unused Production sessions (PEI Function)
- Deletion of entities not associated to a specific use, determined by the unused-entity extraction utility, EXPU (see the PACX procedure in the Manual 'Batch Procedures : User's Guide').

Execution conditions

If the database is available, it may remain open during reorganization since the procedure operates on sequential images of the database.

Updates executed after the back-up file used for reorganization has been built will be retrievable while the reorganized database is being restored.

Batch procedure access authorization option: Global authorization level 4 is required.

Abnormal executions

Refer to Chapter 'Overview', Subchapter 'Abnormal executions'

As specified in paragraph Important recommendations below, the Reorganization procedure can be very long. It is therefore advisable to keep all temporary files after each step.

If one of the steps abends, the procedure can be restarted at the step level, but not at the procedure level.

REOR: Inputs - Processing - Results

A '*' line with user code and password.

Specific user input for the procedure (optional), specifying

- libraries to be purged,
- sessions to be purged or to be kept,
- users to be purged.
- entities to be purged.

a printed copy of the list of index of the REOR procedure.

Pos.	Len.	Value	Meaning
2	1	'B'	Library purge
3		bbb	Library code(s) up to 23 library codes per line

Maximum number of libraries to be purged: 300.

Pos.	Len.	Value	Meaning
2	1	'V'	Purge frozen sessions
		'S'	Save frozen sessions
			'V' and 'S' lines are not compatible
3		ssss	Session number(s): up to 17 session numbers per line

Maximum number of sessions indicated on the request: 999.

Maximum number of frozen sessions in a database: 7,500.

Pos.	Len.	Value	Meaning
2	1	'U'	Purge user
3		uuuuuuuu	User code(s) up to 9 user codes per line

Pos.	Len.	Value	Meaning
2	1	'E'	Physical purge of entities (transactions provided by EXPU)
3	1 2		Entity Type: .Type .UEO call code (if Type ″$″)
6	6		Code of the entity to be purged (may be a joker code)
12	3		Library code 5 groups of type/code entity/lib. possible per 'E'-type line

A Maximum number of 2,500 occurrences of an entity type is processed by the execution of the REOR procedure. The 'List of 'purged' entities' signals when this limit is reached. In case of a generic request, the entity code must be completed with *'s to make up for six characters. If the code contains six '*', all of the entity's occurences will be deleted.

Pos.	Len.	Value	Meaning
2	1	'D'	Printed copy of the list of index of the REOR procedure
3	1	' '	no report of copies of index
		'1'	report of copies of index

When the system finds an input error, it generates an error message and the procedure is not executed.

Estimating file size

The maximum sizes used during this procedure are based on the sizes of the files in the database before reorganization. The report printed by the preceding SAVE procedure provides all the relevant data:

- NI = number of index file records,
- ND = number of data file records MINUS number of gaps,
- NC = number of primary records on the data file,
- NH = number of 'frozen' (historical account) records from the data file (NH = ND - NC)

These symbols are also detailed in the presentation of each of the files for this procedure.

Printed output

This procedure prints a report listing errors found during reorganization, and statistics on the contents of the database.

Results

The output of this procedure is a reorganized sequential image of the database (where purges may have been performed). It does not contain gaps. Gaps can be added by the REST procedure.

Note:

This procedure does not increment the current session number of the database.

Important recommendations

The Reorganization procedure presents a number of characteristics which the user should be aware of:

- The step that rebuilds the Index file uses a large amount of CPU time.
- If the database contains a large amount of data, it is recommended to catalog the temporary files, or to use tape files to obtain the checkpoints in case of an abend in one of the steps.
- If files are transferred onto tape it is preferable to check on the initial blocking factors.
- The space allocated to the sortworks should also be calculated with care.

REOR: Description of Steps

Validation of user input: PTU2CL

This step validates user input and sets a return code when an error is detected.

Code		Type	Label
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database index
PACGGR		Input	Administration Database data
PACGGU		Input	Administration Database users
PAC7PC		Input	Sequential image of the development Database
PAC7MB		Input	Input working file
PAC7BM		Output	Formatted records
PAC7PU		Output	Entity records to be purged (length=44)
PAC7EE		Report	Control report
PAC7DD		Report	Batch procedures authorization option

. Return codes:

- 0: OK
- 4: Error on user input

- 8: No batch procedure authorization

Retrieval of data: PTU200

This step selects 'data' type information in the initial sequential file of the database (in case the Dispatch option is used, it leads to the recognition of one file, that which contains the data, i.e. PC(0)). It then formats the key of each record selected for the subsequent sort.

Code		Type	Label
PAC7AE		Input	Error messages
PAC7BM		Input	User transactions
PAC7PC		Input	Sequential image of the development Database
PAC7PY		Input	Sequential image of the extension data of the development Database
PAC7PR		Output	Formatted records (length=176 size= ND)
PAC7NX		Output	Long data
PAC7NY		Output	Unformatted data
PAC7AU		Output	PR image (length=153)
PAC7EE		Report	Report on statistics retrieval

Tri ASCII : PTU205

Code		Type	Label
PAC7PR		Entrée	Sorted records
PAC7RP		Sortie	ASCII sorted records

The SORT programs requires disk space equivalent to twice the size of the file to be sorted.

Purge: PTU210

This step purges all libraries and sessions entered in the user input. When there is no input, it formats the records.

Code		Type	Label
PAC7AE		Input	Error messages
PAC7PR		Input	Sorted records
PAC7BM		Input	User transactions
PAC7PU		Input	Entity records to be purged
PAC7QS		Output	Purged records (length=176, size= ND)
PAC7UM		Output	Macro-structure call lines (length=176)
PAC7EE		Report	Library and session purge report
PAC7EK		Report	Entity-purge report
PAC7EB		Report	Technical report

Return codes: 0: OK 8: Overload of capacity

The following steps are executed only if the return code is 0.

Index rebuilding: PTU220

This step executes two types of procedures:

- Reconstruction of the indexes using the data

Code		Type	Label
PAC7AE		Input	Error messages file
			User transactions
PACGGN		Input	Administration Database index
PACGGR		Input	Administration Database data
PACGGU		Input	Administration Database users
PAC7UR		Input	Purged data
PAC7NX		Input	Complementary Long data
PAC7UM		Input	Macro-structure call lines
PAC7PA		Output	Data from frozen sessions (length=153 size=NH)
PAC7PB		Output	Data from current session (length=153 size=NC)
PAC7PC		Output	First data records (length=153)
PAC7AN		Output	Permanent index file (length=60 size=NI)
PAC7MR		Input/Output	Macro-structure call lines
PAC7EE		Report	Index-building report

Tri ASCII : PTU205

Code		Type	Label
PAC7AN		Input	Sorted index
PAC7NA		Output	ASCII Sorted index

The SORT programs requires disk space equivalent to twice the size of the file to be sorted.

Extension data processing: PTU226

Code		Type	Label
PAC7NY		input	Unformatted data
PAC7PA		input	Data from frozen sessions
PAC7PB		input	Data from current session
PAC7PC		input	First data records
PAC7QA		output	Data from frozen sessions (length=153)
PAC7QB		output	Data from current session (length=153)
PAC7QC		output	First data records (length=153)
PAC7QY		output	Long data (length=1,018)

Merge: PTU240

This step reconstructs the final sequential image using the temporary files produced by the previous step.

Code		Type	Label
PAC7AE		Input	Error messages
PAC7AN		Input	Sorted index
PAC7AU		Input	PR image
PAC7BM		Input	User transactions
PAC7PA		Input	Data from frozen sessions
PAC7PB		Input	Data from current session
PAC7PC		Input	First data records
PAC7QY		Input	Extension data
PAC7CP		Output	Sequential image of the development Database
PAC7PD		Output	Sequential image of the development Database
PAC7PY		Output	Sequential image of the development Database
PAC7IE		Report	Logical database building

REOR: Execution Script

```
REM * -----------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -----------------------------------------------------
REM *                - REORGANIZATION OF THE DATABASE -
REM *
REM * -----------------------------------------------------
REM *
REM * THE REOR PROCEDURE MAY BE USED IN TWO CASES:
REM * . WHEN PART OF THE DATA WAS DELETED BECAUSE OF A MAL-
REM *  FUNCTION OR SYSTEM FAILURE, AND NO OTHER PROCEDURE CAN
REM *  BE USED (IN PARTICULAR, DELETION OF THE AN INDEX FILE)
REM * . WHEN THE DATABASE IS TO BE PURGED OF THE FOLLOWING:
REM *   - OBSOLETE LIBRARIES AND/OR SESSIONS;
REM *   - ENTITIES NOT USED IN THE DATABASE;
REM *
REM * -----------------------------------------------------
REM *
<job id=REOR>

<script language="VBScript">
MyProc = "REOR"
</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" , "PTU2CL"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7BM") = Rep_TMP & "\WBM.tmp"
WshEnv("PAC7PU") = Rep_TMP & "\WPU.tmp"
WshEnv("PAC7DD") = Rep_USR & "\REORDD2CL.txt"
```

```
WshEnv("PAC7EE") = Rep_USR & "\REOREE2CL.txt"
WshEnv("PAC7MB") = Fic_Input
WshEnv("PAC7PC") = Rep_SAVE & "\PC"
Return = WshShell.Run("BVPTU2CL.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTU2CL")

Call Msg_Log (Array("1022" , "PTU200"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7PC") = Rep_SAVE & "\PC"
WshEnv("PAC7PY") = "NUL"
' WshEnv("PAC7PY") = Rep_SAVE & "\PCY"
WshEnv("PAC7BM") = Rep_TMP & "\WBM.tmp"
WshEnv("PAC7PR") = Rep_TMP & "\WPR.tmp"
WshEnv("PAC7NX") = Rep_TMP & "\WNX.tmp"
WshEnv("PAC7NY") = Rep_TMP & "\WNY.tmp"
WshEnv("PAC7AU") = Rep_TMP & "\WAU.tmp"
WshEnv("PAC7EE") = Rep_USR & "\REOREE200.txt"
Return = WshShell.Run("BVPTU200.EXE" , 1, TRUE)
If Return = 8 Then
Call Msg_Log (Array("1027"))
End If
Call Err_Cod(Return , 0 , "PTU200")

Call Msg_Log (Array("1022" , "PTU205"))
'-----------------------------------
WshEnv("PAC7PR") = Rep_TMP & "\WPR.tmp"
WshEnv("PAC7RP") = Rep_TMP & "\WRP.tmp"
Return = WshShell.Run("BVPTU205.EXE" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTU205")
'OK :
Call DelFile (Rep_TMP & "\WPR.tmp")

Call Msg_Log (Array("1022" , "PTU210"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7EB") = Rep_USR & "\REOREB210.txt"
WshEnv("PAC7EE") = Rep_USR & "\REOREE210.txt"
WshEnv("PAC7EK") = Rep_USR & "\REOREK210.txt"
WshEnv("PAC7BM") = Rep_TMP & "\WBM.tmp"
WshEnv("PAC7PR") = Rep_TMP & "\WRP.tmp"
WshEnv("PAC7PU") = Rep_TMP & "\WPU.tmp"
WshEnv("PAC7QS") = Rep_TMP & "\WQS.tmp"
WshEnv("PAC7UM") = Rep_TMP & "\WUM.tmp"
Return = WshShell.Run("BVPTU210.EXE" , 1, TRUE)
If Return = 4 Then
Call Msg_Log (Array("1057"))
End If
If Return = 8 Then
Call Msg_Log (Array("1056"))
End If
Call Err_Cod(Return , 0 , "PTU210")
'OK :
Call DelFile (Rep_TMP & "\WRP.tmp")
Call DelFile (Rep_TMP & "\WPU.tmp")

Call Msg_Log (Array("1022" , "PTU220"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7BM") = Rep_TMP & "\WBM.tmp"
WshEnv("PAC7AN") = Rep_TMP & "\WAN.tmp"
WshEnv("PAC7MR") = Rep_TMP & "\WMR.tmp"
WshEnv("PAC7NX") = Rep_TMP & "\WNX.tmp"
WshEnv("PAC7PA") = Rep_TMP & "\WPA.tmp"
```

```
WshEnv("PAC7PB") = Rep_TMP & "\WPB.tmp"
WshEnv("PAC7PC") = Rep_TMP & "\WPC.tmp"
WshEnv("PAC7UM") = Rep_TMP & "\WUM.tmp"
WshEnv("PAC7UR") = Rep_TMP & "\WQS.tmp"
WshEnv("PAC7EE") = Rep_USR & "\REOREE220.txt"
Return = WshShell.Run("BVPTU220.EXE" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTU220")

Call Msg_Log (Array("1024"))
'-----------------------------------
Call DelFile (Rep_TMP & "\WUM.tmp")
Call DelFile (Rep_TMP & "\WQS.tmp")

Call Msg_Log (Array("1022" , "PTU225"))
'-----------------------------------
WshEnv("PAC7AN") = Rep_TMP & "\WAN.tmp"
WshEnv("PAC7NA") = Rep_TMP & "\WNA.tmp"
Return = WshShell.Run("BVPTU225.EXE" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTU225")

Call Msg_Log (Array("1024"))
'-----------------------------------
Call DelFile (Rep_TMP & "\WAN.tmp")

Call Msg_Log (Array("1022" , "PTU226"))
'-----------------------------------
WshEnv("PAC7PA") = Rep_TMP & "\WPA.tmp"
WshEnv("PAC7PB") = Rep_TMP & "\WPB.tmp"
WshEnv("PAC7PC") = Rep_TMP & "\WPC.tmp"
WshEnv("PAC7QA") = Rep_TMP & "\WQA.tmp"
WshEnv("PAC7QB") = Rep_TMP & "\WQB.tmp"
WshEnv("PAC7QC") = Rep_TMP & "\WQC.tmp"
WshEnv("PAC7QY") = Rep_TMP & "\WQY.tmp"
WshEnv("PAC7NY") = Rep_TMP & "\WNY.tmp"
Return = WshShell.Run("BVPTU226.EXE" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTU226")

Call Msg_Log (Array("1024"))
'-----------------------------------
Call DelFile (Rep_TMP & "\WPA.tmp")
Call DelFile (Rep_TMP & "\WPB.tmp")
Call DelFile (Rep_TMP & "\WPC.tmp")

Call Msg_Log (Array("1022" , "PTU240"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_TMP & "\WNA.tmp"
WshEnv("PAC7AU") = Rep_TMP & "\WAU.tmp"
WshEnv("PAC7BM") = Rep_TMP & "\WBM.tmp"
WshEnv("PAC7PA") = Rep_TMP & "\WQA.tmp"
WshEnv("PAC7PB") = Rep_TMP & "\WQB.tmp"
WshEnv("PAC7PC") = Rep_TMP & "\WQC.tmp"
WshEnv("PAC7QY") = Rep_TMP & "\WQY.tmp"
WshEnv("PAC7CP") = Rep_SAVE & "\PC.new"
WshEnv("PAC7PD") = Rep_SAVE & "\PCI.new"
WshEnv("PAC7PY") = Rep_SAVE & "\PCY.new"
WshEnv("PAC7IE") = Rep_USR & "\REORIE240.txt"
Return = WshShell.Run("BVPTU240.EXE" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTU240")

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

If Return = 0 then
Call Msg_Log (Array("1022" , "BACKUP"))
'-----------------------------------
```

```
Call Turnover(Rep_SAVE & "\PC")
Call Turnover(Rep_SAVE & "\PCI")
Call Turnover(Rep_SAVE & "\PCY")
End if

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

</script>
</job>
```

Chapter 4. Manager's Utilities

PACX: 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: Inputs Common to 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

Pos.	Len.	Value	Meaning
		'N'	For RMEN only : no extraction of locked entities by an other user
50	1	' '	No transfer of password
		'1'	Password transfer
69	3	bbb	Library code for the '*'-line of the output file(s) (For EXTR,EXLI, and EXUE only)
76	5	nnnnT	Session number for the '*'-line of the output file(s) (For EXTR,EXLI, and EXUE only)

(1) 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 Archived Transactions

EXPJ: Introduction: The EXPJ procedure has a two-fold action:

- It converts the Journal file into update transactions with possible selection from a range of dates, sessions, libraries, etc.
- It prints out a listing of the contents of the archived Journal file, using the same criteria.

Its main purpose is to retrieve transactions associated with one database in order to update another database.

It is executed on the archived Journal file (PJ).

Execution conditions

none.

EXPJ: User Inputs: User entry specific to this procedure and specifying the extraction characteristics.

Pos.	Len.	Value	Meaning
2	1	'J'	Line code
3	1	'S' 'D'	Selection on session number Selection on date
4	1	' ' 'N'	Chronological sort No chronological sort
5	1	' ' 'N'	Sort by user No sort by user
6	1	' ' 'N'	Sort by Library No sort by library
7	1	' ' 'N'	Sort by session No sort by session
8	8	uuuuuuuu	User code for batch update
16	8	pppppppp	User password
24	4	dddd	Session number: beginning (if 'S')
28	4	ffff	Session number: end (if 'S')
32	8	CCYYMMDD	Date of beginning of select.(if 'D')
40	8	CCYYMMDD	Date of end of selection (if 'D')
48	1	' ' 'Z' 'T'	Version of selected transactions Selection of all sessions Selection of current session Selection of frozen session
49	3	'bbb'	Code of selected library
52	5	'ssssT'	Selection of T-type session (test version of frozen session:'ssssT')
57	3	ppp	DSMS Product Code
60	6	nnnnnn	DSMS Change number (Selection by change number-DSMS)
66	6	HHMMSS	Starting time
72	6	HHMMSS	Ending time
80	1	'*'	If selection on user, indicates following line

Second user entry if selection on user code :

Pos.	Len.	Value	Meaning
2	1	'J'	Line code
3	1	'*'	Indicates a following line
4	8	uuuuuuuu	User code

Reports

- The list of selection options used,
- The list of selected transactions, if requested.

Result

In the case of a request for conversion of the Journal entries into transactions, the result of the EXPJ procedure is a sequential file containing all selected transactions.

Extraction of Libraries

EXLI: Introduction: The EXLI procedure extracts a complete library from the database and transforms it in transactions which are used in the update or comparison procedures.

The file obtained, according to its formatting, can be used as input to the UPDT, UPDP or CPSN procedures.

Execution conditions

If DESIGN entities have been downloaded and have then been locked, they must be uploaded before the extraction to ensure data consistency.

EXLI: Inputs: No specific line, but as many '*'-lines as there are libraries to be extracted in the sub-network.

Printed output

The extractor prints:

- A list of extracted libraries with the number of records for each library,
- The details of records extracted for each library.

Extraction for Purge

EXPU: Introduction: The EXPU utility purges: unused entities from a database, frozen sessions which have been logically cancelled, cancelled libraries, GP lines belonging to the users who no longer exist.

Several types of purges are possible:

- 'Logical' purge of entities which have become obsolete;
- 'Logical' purge of unsued entities in a particular context;
- 'Physical' purge of entities which have never been used.

Terminology

Final entities:

These entities, which are not used by other entities.

Free-type cross-reference:

Reference whose existence does not prevent deletion of the Definition screen of the Entity on which it is dependent.

Principles

Logical purge: the EXPU procedure shows the list of entities which have not been used since an indicated frozen session and in a given context.

For these entities, the procedure generates logical deletion transactions of definition and description lines. These transactions can be used as input to the UPDT procedure.

For free-type entities, no deletion transaction is generated: only a message is printed in the report.

- Physical purge: the EXPU procedure gives the list of entities which have never been cross-referenced since they have been created in a given context. For these entities, physical purge transactions are generated. They can be used as input to the REOR procedure.

  Limits of physical purge: if a data structure has already been purged, there is no physical purge possible for its segments.
- Purge of frozen sessions: The extraction for purge procedure indicates to the user the sessions which have been logically cancelled.

  For these sessions, physical-purge transactions are generated. They can be used as input to the reorganization procedure.
- Purge of libraries: The extraction for purge procedure indicates to the user the libraries which have been cancelled.

  For these libraries, physical-purge transactions are generated. They can be used as input to the reorganization procedure.
- Purge of GP lines: The extraction for purge procedure specifies to the user, the users who no longer exist in the administration database, despite the GP lines which still exist in the database.

  For these users, physical-purge transactions are generated. They can be used as input to the reorganization procedure.

Execution conditions

None.

EXPU: Inputs: One line with the extraction characteristics:

Pos.	Len.	Value	Meaning
2	2	'P '	Line code
3	1		Purge of frozen sessions
		'S'	yes
		' '	no purge
4	1		Purge of libraries
		'B'	yes
		' '	No purge
5	1		Purge of GP lines for users who no longer exist
		'U'	yes
		' '	No purge
6	1		Purge of entities
		'P'	Physical (via the REOR procedure)
		'L'	Logical (via the UPDT/UPDP update)
		' '	No purge
7	1	t	Entity type
8	1	m	Print option for the last entity update: user code and date

Pos.	Len.	Value	Meaning
9	1	s	Print option for the last session in which the entity is used
10	4	ssss	Session number (L type only) from which the entities should not be used in order to be logically purged
14	6	pppppp	Program code where the search stops if the programs are processed Information required if the entity type is 'P' or if it is blank

Comments

Each entity type may be processed separately. If there is no entity type entered, all the entities are processed except the final entities.

Command examples:

*user____passwordLIB

P___PE1

Command for physical purge transactions for the data elements in the BIB library sub-network and printing of the last update (user and date).

*user____passwordLIB

P___LP112222PROGR

Command for logical deletion transactions for the programs in the BIB library sub-network whose codes are less than or equal to PROGR, and no longer used since the 2222 session with the printing of the last update (user and date) and the last session of use.

*user____passwordLIB

PSBUP_____PROGR

Command for the physical purge transactions for all entities in the BIB library sub-network (except the final entities), for logically cancelled frozen sessions, for deleted libraries and GP lines of users who no longer exist.

Printed output

This procedure prints out:

- The list of entities to be purged logically,
- The list of entities to be purged physically,
- The list of entities copied in the sub-network,
- The list of frozen sessions to be purged physically.
- The list of libraries to be purged physically.
- The list of users whom GP lines are to be purged physically.

Result

The result of this procedure is:

- In the case of a logical purge, a sequential file containing entity deletion transactions to be used as input in the batch updating UPDT or UPDP procedures. These transactions are sorted as follows:
  - By decreasing hierarchical library level.
  - By library.
  - By record type: descriptions, definition screens.
- In the case of a physical purge, purge of frozen sessions, purge of libraries or GP lines, a sequential file of purge transactions to be used as input in the REOR procedure.

  Each transaction contains an entity to be purged: For each entity, the following information is included:
  - The type of entity.
  - The entity code.
  - The library code (see section ″Inputs-recommandations″ in sub-chapter ″Database reorganization″).

Standardization Utility

RMEN: Introduction: RMEN: ENTITY RENAMING / MOVING - INTRODUCTION

The RMEN procedure is an optional utility. It is subject to a separate purchase agreement.

Through this procedure you can:
- Rename an entity
- Replace an entity with another
- Move an entity to a higher-level library
- Rename and move up an entity simultaneously.

This procedure may be applied to Dictionary and entities.

Its output is a file containing update transactions, which will be used as input to the batch update procedure (UPDT or UPDP).

Execution conditions

None, since the Database is not directly updated.

RMEN: Inputs: Several command lines per entity to be processed:

First line - concerned entity :

Pos.	Len.	Value	Meaning
2	2	'W2'	Line code
4	2		Processing option:
		'MV'	Entity move (UP)
		'RN'	Entity rename
		'MR'	Upward move and rename
		'RP'	Entity replace
6	1	' '	Line type

Pos.	Len.	Value	Meaning
7	3	ttt	Entity type or local code of a WorkStation entity: D, E, I, O, P, R, S, T, $nn, Ynn,M Q, B, V, or DST, DEL ...
10	30	eeee..	Code of entity to be extracted

Second line - environment :

Pos.	Len.	Value	Meaning
2	2	'W2'	Line code
4	2		Processing option - same as line 1
6	1	'E'	Line type
7	3	sss	Source library code (for MOVE)
10	3	'//A' '//M' '//D' '//O' '//F'	For extraction of WorkStation enti- ties: methodology code SSADM MERISE YSM OMT IFW
13	3	'ALL'	for 'MV' and 'MR': Selects all occu- rrences of a UE or all Segments or Reports of a Data Structure (implicit option for 'RN' and 'RP)
16	6	rrrrrr	Parent Data Element code

Third line - new codes:

Pos.	Len.	Value	Meaning
2	2	'W2'	Line code
4	2		Processing option - same as line 1
6	1	'N'	Line type
7	30	nnnn...	New entity code
37	8	gggggggg	For programs and screens, new generated code
45	6	cccccc	For programs, new sequencing code
51	8	eeeeeeee	For screens, new map code

Fourth line - selection for REPLACE:

Pos.	Len.	Value	Meaning
2	2	'W2'	Line code
4	2	'RP'	'REPLACE'
6	1	'S'	Line type
7	3		REPLACE: Selection of the types of the entities to be modified 'DEL': Data Element 'DBD': Database Block 'DST': Data Structure 'SEG': Segment 'RPT': Report 'TXT': Text 'VOL': PDM volume 'PGM': Program 'SCR': Screen 'PIA': P.I.A. 'MET': Methodology 'CME': Client Meta-Entity 'CRL': Client Relationship '$tt': Client User Entity : (tt = type code) '$**': All Client User Entities 'Ytt': Extension User Entity : (tt = type code) 'Y**': All Extension User Entities
10	30		REPLACE: Codes of entities to be modified (* may be used if you want to specify only the beginning of a code.

Lines for REPLACE (continuation lines for selection):

Pos.	Len.	Value	Meaning
2	2	'W2'	Line code
4	2	'RP'	'REPLACE'
6	1	'*'	Line type
7	3		Selection of types of entities to be modified
10	30		Codes of entities to be modified

Last line (required):

Pos.	Len.	Value	Meaning
2	2	'W2'	Line code
4	2		Processing option - same as line 1
6	1	'/'	Line type

Request-sequencing requirements

The sequencing of RMEN requests should follow a logical order.

Examples:

- A parent Data Element must be moved to the higher-level library before its child Data Element(s).
- When a Segment is called by another Segment, the called Segment must be moved to the higher-level library before the Segment that is calling it.
- When a macro-structure is called by a batch Program or on-line Screen, it must be moved into the higher-level library before this Program or Screen.

Request-input requirements

All input is required except:

- The source library code in case of entity renaming (RN) or replacing (RP),
- The new entity code in case of upward move (MV),
- The code of the parent data element (except when a child data element is to be associated with it).

  The 'RP' processing type is incompatible with the other processing types.

Execution rules

The source library must belong to the sub-network of the target library.

When an upward move is requested for an entity which already exists in the target library, a warning message appears in the report, but the transaction is still generated.

Printed output

This procedure prints out the following:

- The list of entities processed by RMEN.
  - The number of lines extracted for each request.

Result

The output is a sequential file which contains update transactions:

- Creation or modification transactions sorted by:
  - Ascending library hierarchical level,
  - Library,
- Record type (uses, definition, or description).
  - Deletion transactions sorted by:
  - Descending library hierarchical level,
  - Library,
  - Record type (uses, description, definition).

Notes:

The replacement of entities (RP) does not ensure data consistency.

Example: if you replace a Data Element with another one in a Segment, RMEN does not modify the program lines where this Data Element is used by this Segment, except if you have requested the replacement in programs.

New occurrence codes longer than the initial ones may sometimes cause update transactions to be truncated. However, they will still belong to the flow of update transactions, but will also appear in the validation report with a warning message.

If not correctly managed, the RMEN procedure may have undesired effects on the Database. Caution is highly recommended when requesting its execution.

RMEN: Recommendations and Restrictions: Processing in a frozen session is possible. The number of the session is indicated on the '*' line.

When an error is detected on the '*' line, the request flow is not processed.

Only one '*' line is authorized.

All entity types

The MOVE+RENAME (MR) command first moves and then renames. The consequence is that all the entities bearing the same code within the sub-network of libraries equal to or lower than the target library are renamed by the RMEN procedure.

If this result is not satisfactory, it is advised to first run a RMEN/RENAME followed by a UPDT, then a RMEN/MOVE followed by another UPDT execution.

- If the entity uses other entities, these entities used must exist in a library whose level is greater or equal to that of the target library.
- When an occurrence is renamed, if it is called on Assigned Text (-AT) lines,
  - it is changed on I-type lines,
  - it is not changed on J-type lines.

Data structures

Renaming a Data Structure causes the renaming of all its Segments and Reports.

Caution :

An upward move of a Data Structure involves the upward move of all of its Segments and Reports contained in the source library in cases where the 'global upward move' field contains 'ALL'. If this field is blank, the Segments and Reports remain in the source library.

The existence of the Data Structure in an upper-level library is checked.

Segments

These entities can only be moved upward. Their Data Structure must exist in a library whose level is higher than or equal to that of the target library.

The existence of a Segment in a library whose level is higher than or equal to that of the target library is checked, as is that of called Segments, Data Elements, and MERISE Objects and Relationships.

Reports

It is not possible to change a single report code or to replace a singlr report. It is however possible to perform an upward move of a single report.

However, you can rename, move upward or replace all the reports with the same prefixes (two first characters), you just need to enter '*' in the third character place: W2RN R xx* or W2MV R xx* or W2MR R xx* or W2RP R xx*

An existence validation is performed in a library upper or equal to the target library for the called data elements only.

Data elements

The indication of a parent Data Element code affects only the Data Element Definition in the source library. By default, a child Data Element remains attached to its parent. However, it is possible to suppress this link by entering the code '&&&&&&&' in the parent Da Element field.

A child Data Element can be turned into a parent Data Element or may be assigned another parent by specifying a parent Data Element code. This parent Data Element must be defined in a library upper or equal to the target library.

A parent Data Element contained in a request must not have been previously processed as a source Element.

The format of the Data Element being moved remains the same, whatever the modification in relation to a parent Data Element.

If the target Data Element is used as an undefined Data Element, the format of its uses (on Segment or Report '-CE' screens) must correspond to the format specified in the Definition.

The renaming of a key Data Element of a Data Structure (indicated as an argument on the Call of Data Structures '-CD' screen) is not allowed.

Programs

Their processing goes through a check on libraries whose level is higher than or equal to that of the target library of :

- Macro-Structures,
- Data Structures,
- Segments or Data Elements (called in WORKING-STORAGE).

Screens

Screens are processed individually. RMEN does not process the whole Dialogue. The Dialogue must therefore exist in a library whose level is higher than or equal to that of the target library.

Meta-entities

A Meta-entity can be processed only if there is no other meta-entity bearing the same call code in the sub-network of the target library.

Caution:

When the global upward move field contains 'ALL', an upward move of a User Entity involves the upward move of all of its occurrences contained in the source library. If this field is blank, the occurrences remain in the source library.

The existence of all Data Elements and User Relations called in the Definition lines is checked in a library higher or equal to the target library.

User entity

The existence of the Meta-entity in a library higher or equal to that of the target library is checked, as is that of occurrences linked to the UEO or details lines.

MERISE entities

For MERISE Objects and Elements/Properties called in description screens, an existence check is performed in the library whose level is higher than or equal to that of the target library.

Database blocks

The existence of MERISE objects or called segments is checked.

Volumes

The existence of Reports called in the Volume Definition screen is checked.

The WorkStation entities

Calls of the '//M', '//Y' and '//D' type are used to extract all the WorkStation entities. The local entity type must be entered (in the entity type field) as well as the code of entity before processing, the library code and the code of the entity after processing. The WorkStation methodology (MERISE, IFW, OMT, YSM...) is entered in a special field at position 10 in the command line corresponding to the environment (line type : 'E').

Caution: One procedure execution can process occurrences related to only one Methodology.

Sub-Network Comparison

CPSN: Introduction: The Sub-Network Comparison procedure (CPSN) compares the images of two sub-networks extracted by the PACX procedure (EXLI extractor, formatting for CPSN), which may or may not belong to the same network, or the images of entities extracted by the PACX procedure (EXTR or EXUE extractor, formatting for CPSN), in order to obtain the batch update transactions which will align the 'slaves' sub-network or entities with the 'masters' sub-network or entities.

- 'Master' sub-network = reference sub-network,
- 'Slave' sub-network = sub-network to align with the Master entity.
- 'Master' entity = reference entity,
- 'Slave' entity = entity to align with the

Execution condition

None.

Abnormal executions

If an abend occurs, the procedure can be restarted as it is once the problem has been solved.

CPSN: Inputs: No spécific line.

Notes

The sub-networks or entities to be compared must have been extracted via the PACX procedure (EXLI, EXTR or EXUE extractor; formatting for CPSN).

They must contain the same number of libraries (checked by the system) and have the same structure.

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
PAC7MA		Input	CPSN Master file

Code		Type	Label
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>
```

Session Management

Introduction

The VA Pac session number cannot be greater than 9999. When the session number is close to 9999, the utility program re-assigns all the session numbers by incrementing the numbers of frozen sessions by 1 (starting from session 0001 or from a session chosen by the Administrator).

The utility consists in two procedures: the ESES preparation procedure and CSES compression procedure.

Note: The session freeze is performed by the UPDT procedure. It increments the current session number.

This reassignment is carried out on sequential images of the files that include the session number, i.e. the backup files of the Database (PC), of the Journal (PJ), of the DSMS Journal (BJ), of the DSMS Database (BB), and of the Pactables Database (TC).

This utility includes two procedures: ESES and CSES.

ESES: Session Numbers Extraction

ESES: Introduction

The Extraction of Session Numbers procedure (ESES) creates a correspondence-table file linking older frozen sessions and new frozen sessions.

Preliminary operations

Backup of the VA Pac files:

- Archival of the Journal (ARCH)
- Backup of the VA Pac Database (PACS SAVE option)

If Pactables is installed:

- Table backup (SVTA)

If DSMS is installed, perform a backup of the DSMS environment, by:

- Archiving the DSMS Journal (DARC)
- Backing up the DSMS Database (DSAV)

EXECUTION CONDITIONS

None.

ESES: Inputs

A '*' line with User code and Password is required.

One line per session number to force :

Pos.	Lon.	Valeur	Meaning
2	1	'S'	Line Code
3	4	nnnn	Original session number
7	4	nnnn	New session number

ESES: Description of Steps

Input recognition: PTU001

Creation of the session-number correspondence file: PTUESS

Code		Type	Label
PAC7AE		Input	Error messages
PAC7AR		Input	Development Database data
PAC7AN		Input	Development Database index
PACGGR		Input	Administration Database data
PACGGN		Input	Administration Database index
PACGGU		Input	Administration Database users
PAC7MB		Input	Input transcations
PAC7MV		Output	Session-number correspondence table
PAC7EU		Report	Extraction report
PAC7DD		Report	Batch procedure authorization option

Return code: 8: Unauthorized user.

ESES: Execution Script

```
REM * -------------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -------------------------------------------------------
REM *         - SESSION NUMBERS CORRESPONDENCE TABLE -
REM *
REM * -------------------------------------------------------
REM *
REM * THE EXTRACTION OF SESSION NUMBERS PROCEDURE
REM * (ESES) CREATES A CORRESPONDENCE-TABLE FILE LINKING
REM *  OLDER FROZEN SESSIONS AND NEW FROZEN SESSIONS.
REM *
REM * INPUT :
REM *  -  USER IDENTIFICATION LINE (REQUIRED)
REM *  -  COMMAND LINE :
REM *  COL 2  : 'S'    LINE CODE
REM *  COL 3  : (4 N)  ORIGINAL SESSION NUMBER
REM *  COL 7  : (4 N)  NEW SESSION NUMBER
REM * -------------------------------------------------------
REM *
<job id=ESES>

<script language="VBScript">
MyProc = "ESES"
</script>
```

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

<script language="VBScript">

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

If MyUser = "" then
Rep_RI = Rep1_USR
Else
Rep_RI = Rep1_USR & "\" & MyUser
End If

Call Msg_Log (Array("1022" , "PTUESS"))
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7MB") = Fic_Input
WshEnv("PAC7MV") = Rep_RI & "\Mveses.tmp"
WshEnv("PAC7DD") = Rep_USR & "\ESESDDESS.txt"
WshEnv("PAC7EU") = Rep_USR & "\ESESEUESS.txt"
Return = WshShell.Run("BVPTUESS.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTUESS")

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

</script>
</job>
```

CSES: Compression of Session Numbers

CSES: Introduction

The Compression of Session Numbers procedure (CSES) compresses the session numbers of the development Database logical backups, the Pactables Database if this module is installed on the site, and the DSMS Database if this module is installed on the site. It uses the correspondence table created by the ESES procedure.

The resulting files must be restored.

Execution conditions

None.

Yet, all the backups to be processed must be valid.

CSES: Inputs

A * line with User Code and Password.

The user input is used to indicate the list of files to be retrieved (PC, PJ, BB, BJ, and TC), in order to execute the retrieval after one or several runs.

The line is built as follows:

Col.	Len.	Value	Meaning
2	1	'S'	Line code

Col.	Len.	Value	Meaning
3	21		Code of the files to retrieve (PC PJ BB BJ TC) separated with a blank
33	4		If the DSMS database has to be retrieved: development Database logical code

CSES: Description of Steps

Input recognition: PTU001

Compression of session numbers: PTUCSS

Code		Type	Label
PAC7AE		Input	Error messages
PACGGR		Input	Administration Database data
PACGGN		Input	Administration Database index
PACGGU		Input	Administration Database users
PAC7MV		Input	Session numbers correspondence table
PAC7MB		Input	Parameter line
PAC7PC		Input	Development Database backup
PAC7PD		Input	If Dispatch option of the backup: Backup 2 of the development Database
			If Dispatch option of the backup: Backup 3 of the development Database
PAC7CP		Output	If Dispatch option of the backup:
PAC7DP		Output	If Dispatch option of the backup: Backup 2 of the development Database
			If Dispatch option of the backup: Backup 3 of the development Database
PAC7PJ		Input	Backup of the development Dababase Journal
PAC7JP		Output	Backup of the development Dababase Journal
PACDBB		Input	DSMS Database backup (if DSMS is installed)
PACDJB		Output	DSMS Database backup (if DSMS is installed)
PACDDJ		Input	Retrieval of DSMS archived Journal (if DSMS is installed)
PACDJD		Output	Retrieval of DSMS archived Journal (if DSMS is installed)
PAC7TC		Input	Retrieval of Pactables backup (if Pactables is installed)
PAC7CT		Output	Retrieval of Pactables backup (if Pactables is installed)
PAC7EU		Report	Execution output
PAC7DD		Report	Batch procedure authorization option

CSES: Execution Script

```
REM * -----------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -----------------------------------------------------
```

```
REM *             - COMPRESSION OF SESSION NUMBERS -
REM *
REM * ------------------------------------------------------
REM *
REM * THE COMPRESSION OF SESSION NUMBERS PROCEDURE (CSES)
REM * COMPRESSES THE SESSION NUMBERS OF THE DEVELOPMENT
REM * DATABASE LOGICAL BACKUPS, THE PACTABLES DATABASE IF
REM * THIS MODULE IS INSTALLED ON THE SITE, AND THE DSMS DATA
REM * BASE IF THIS MODULE IS INSTALLED ON THE SITE. IT USES
REM * THE CORRESPONDENCE TABLE CREATED BY THE ESES PROCEDURE.
REM * THE RESULTING FILES MUST BE RESTORED.
REM *
REM * INPUT :
REM *  -  USER IDENTIFICATION LINE (REQUIRED)
REM *  -  COMMAND LINE :
REM *  COL 2  : 'S'        LINE CODE
REM *  COL 3  : (21 CAR.)  CODE OF THE FILES TO RETRIEVE (PC
REM *            PJ PG PP BB BJ TC) SEPARATED WITH A BLANK
REM * COL 33 : (4 CAR.)   IF THE DSMS DATABASE HAS TO BE
REM *          RETRIEVED : DEVELOPMENT DATABASE LOGICAL CODE
REM * ------------------------------------------------------
REM *
<job id=CSES>

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

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

<script language="VBScript">

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

If MyUser = "" then
Rep_RI = Rep1_USR
Else
Rep_RI = Rep1_USR & "\" & MyUser
End If


Call Msg_Log (Array("1022" , "PTUCSS"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7MB") = Fic_Input
If FSO.FileExists(Rep_RI &  "\Mveses.tmp") Then
WshEnv("PAC7MV") = Rep_RI &  "\Mveses.tmp"
Else
Call Msg_Log (Array("1001" ,"Rep_RI &  "\Mveses.tmp"))
c_error = 1
WshVolEnv("RC") = 32
WshEnv("PAC7MV") = "NUL"
End If
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7PC") = Rep_SAVE & "\PC"
WshEnv("PAC7CP") = Rep_SAVE & "\PC.NEW"
WshEnv("PAC7PD") = Rep_SAVE & "\PCI"
WshEnv("PAC7DP") = Rep_SAVE & "\PCI.NEW"
WshEnv("PAC7PJ") = Rep_SAVE & "\PJ"
WshEnv("PAC7JP") = Rep_SAVE & "\PJ.NEW"
WshEnv("PACDBB") = Rep_SAVE & "\BB"
WshEnv("PACDJB") = Rep_SAVE & "\BB.NEW"
WshEnv("PACDDJ") = Rep_SAVE & "\BJ"
WshEnv("PACDJD") = Rep_SAVE & "\BJ.NEW"
WshEnv("PAC7TC") = Rep_SAVE & "\TC"
```

```
WshEnv("PAC7CT") = Rep_SAVE & "\TC.NEW"
WshEnv("PAC7DD") = Rep_USR & "\CSESDDCSS.txt"
WshEnv("PAC7EU") = Rep_USR & "\CSESEUCSS.txt"
Return = WshShell.Run("BVPTUCSS.exe" , 1, TRUE)
If Return = 8 Then
Call Msg_Log (Array("1027"))
End If
Call Err_Cod(Return , 0 , "PTUCSS")

If Return = 0  and  c_error = 0 then
Call Msg_Log (Array("1022" , "BACKUP"))
'-----------------------------------
Call Turnover(Rep_SAVE & "\PC")
Call Turnover(Rep_SAVE & "\PJ")
Call Turnover(Rep_SAVE & "\BB")
Call Turnover(Rep_SAVE & "\BJ")
Call Turnover(Rep_SAVE & "\TC")
End If

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

</script>
</job>
```

Chapter 5. Analisis of Activity and Quality Control

Analisis of Activity

ACTI: Introduction

The ACTI procedure is an optional utility, and its use depends on the corresponding purchase agreement.

The Specifications Dictionary manages all the data related to the various applications being developed or maintained at the site.

The Journal file contains all the database update transactions. As such, it reflects user activity.

With the Journal Statistics Utility (ACTI), this activity can be monitored and presented in the form of charts.

The Journal Statistics Utility allows the Database Manager to query the Journal backup file based on various parameters:

- library code,
- user code,
- entity type,
- entity code,
- line code,
- transaction type,
- date of update,
- time of update,
- session number of update,
- transaction code,
- change number.

These criteria are used to specify the REQUEST AREA.

Results are obtained in the form of three types of charts, i.e., statistical reports, curve-type graphs, or lists of transactions.

This output will be printed according to the selected PAGE LAYOUT. Statistics and graphs are sorted and calculated according to the user request.

- Output Report Type,
- page layout criteria,
- Request Area,
- Data sequencing mode,
- Activity calculation mode.

Execution conditions

None.

ACTI: Query Language

REQUEST CODING

A Journal Statistics Request consists of five different types of lines, identified by the following KEYWORDS:

```
- OUTPUT    :   Output Report Type,
- PAGE      :   Page Layout (page breaks),
- AREA      :   Request Area,
- LINE      :   Statistical Report Lines,
- COLUMN    :   Statistical Report Columns,
- ABSCISSA  :   Curve-type graph Abscissas,
- ORDINATE  :   Curve-type graph Ordinates.
```

The meaning of the keywords, the parameters which define them, as well as their compatibility are explained in paragraph 'KEYWORDS DEFINITION AND VALUES'.

The OUTPUT line is required; the PAGE and AREA lines are op- tional. The LINE, COLUMN, ABSCISSA, and ORDINATE lines are either required or prohibited, depending on the requested output report type.

Only the first three characters of a keyword are used to identify a line type.

On the printed report, each request line is explicitly stated on the first page and an explicit error message is generated in case of a rejected line.

Request lines must be entered in the following order:

OUTPUT PAGE AREA LINE COLUMN ABSCISSA ORDINATE

Any error in this sequence will be considered as the beginning of another request.

The user may enter up to 10 requests at the same time.

The purpose of the ':' character is to mark the end of the keyword.

The rest of the line contains the parameters of each characteristic.

Parameters

Parameters are used to define page layouts, lines and abscissas. These are called 'Presentation Criteria'.

Parameters followed by '=' and a value are called 'Selection Criteria'.

Parameters which define calculations are called 'Calculations'.

The coding, meaning and compatibility of the parameters are described in paragraph 'Parameters: definition and comments'.

SEPARATORS

The data entered on request lines are separated and grouped together using the following characters:

- ':' = end of keyword,
- '=' = link between a parameter and its value,
- '( )' = set of parameters for calculations,
- ',' = parameter or calculation separator,
- '/' = calculation combination,
- '*' = generic selection,
- 'Blank' = end of line (subsequent data is entered for documentary purposes).

Keywords : Meanind and filling modes

OUT(put) : Output report type

This type of line is required at the beginning of each request.

The parameters used to define the output report type are:
- STA for statistics
- GRA for graph
- LIS for list

PAG(es) : Page layout

This type of line is used to indicate at which level a page skip is to be inserted.

The Page layout line is optional.

Headings are printed for each level, as well as totals for the statistical reports.

The page layout is defined by a series of parameters (three maximum separated by the ',' character) identifying data from the Journal, and called 'presentation criteria'.

Example: A page skip may be requested for each user and for each library.

ARE(a) : Request area

This type of line is used to define the transactions to be taken into account.

The REQUEST AREA line is optional.

The Request Area is defined by parameters (separated by the ',' character) followed by the '=' character and the selected value.

Example: The request applies to only some users and for a given period of time.

LIN(es) : Data sorting mode

or

ABS(cissa)

This type of line is used to define either the lines of a statistical report or the X-axis of a curve-type graph.

It is required for both statistical reports and curve-type graphs. However, it is not permitted for transaction lists.

There may be several lines of this type for statistical report.

The Data Sorting Mode may be defined by Presentation Criteria, as well as Selection Criteria. Parameters and values are separated by the ',' character.

Example: Data is sorted by entity type for a statistical report, or by week for for a curve-type graph.

COL(umns) : Activity calculation mode

or

ORD(inate)

This type of line defines the columns of a statistical report or the ordinates of a curve-type graph (maximum of seven columnsor curves).

It is required for both statistical reports and curve-type graphs. However, it is not permitted for transaction lists.

Each column or curve is determined by a calculation, followed by bracketed Selection Criteria. Columns or curves, parameters and values, are all separated by the ',' character.

A printing character (&Cn*dofHAR='X') must be specified for each curve.

A statistical report column may be defined by the relationship between two calculations; these calculations are separated by the '/' character.

Example: A first column or a first curve may be a calculation of the transactions entered on-line, while a second one may show the ratio between the input transactions and the real transactions.

Parameters: definition and comments

&LIB : Library code

This parameter is used as a Selection Criterion to define the Page Layout, the Request Area, the Data Sorting Mode, and the Activity Calculation Mode.

A generic selection may be requested by simply replacing every appropriate character by the '*' character.

&USER : User code

This parameter is used as a Presentation and Selection Criterion to define the Page Layout, the Request Area, the Data Sorting Mode, and the Activity Calculation Mode.

A generic selection may be requested by simply replacing every appropriate character by the '*' character.

&ENTG : Entity type

This parameter is used as a Presentation and Selection Criterion to define the Page Layout, the Request Area, the Data Sorting Mode, and the Activity Calculation Mode.

&ENTD : Line code / Entuty type

This parameter is used as a Presentation and Selection Criterion to define the Data Sorting Mode.

Values are selected according to the entity type entered in the preceding parameter.

&LICO : Line code

This parameter is used as a Presentation and Selection Criterion to define the Page Layout, the Request Area, the Data Sorting Mode, and Activity Calculation Mode.

Values are selected according to the batch line codes.

&ENT : Entity code

This parameter is used as a Presentation and Selection Criterion to define the Page Layout, the Request Area, the Data Sorting Mode, and the Activity Calculation Mode.

A generic selection may be requested by simply replacing every appropriate character by the '*' character.

Values are selected according to the entity type and code.

&INPT : Input type

This parameter is used as a Presentation and Selection Criterion to define the Page Layout, the Request Area, the Data Sorting Mode, and the Activity Calculation Mode.

The value 'B' corresponds to batch input mode; any other value corresponds to on-line input mode.

&D1 : Starting date

This parameter is used as a Selection Criterion to define the Request Area, the Data Sorting Mode, and the Activity Calculation Mode.

This parameter has to be followed by a date (MMDDCCYY). If this parameter is missing, the starting date coincides with the beginning of the Journal.

&D2 : Ende date

This parameter is used as a Selection Criterion to define the Request Area, the Data Sorting Mode, and the Activity Calculation Mode.

This parameter has to be followed by a MMDDCCYY date format.

If this parameter is missing, the end date coincides with the end of the Journal.

&S1 : Starting session

This parameter is used as a Selection Criterion to define the Request Area, the Data Sorting Mode, and the Activity Calculation Mode.

This parameter has to be followed by a four-character session number. If this parameter is missing, the starting session coincides with the beginning of the Journal.

&S2 : Final session

This parameter is used as a Selection Criterion to define the Request Area, the Data Sorting Mode, and the Activity Calculation mode.

This parameter has to be followed by a four-character session number. If this parameter is missing, the final session coincides with the end of the Journal.

&DAY : Day-by-day presentation

Used as a Presentation Criterion to define the page layout and the data sorting mode.

To define an X-axis, this parameter must be followed by the '=' character and the number of characters corresponding to the curve step (its default value is one character).

&WEEK : Week-by-week presentation

Used as a presentation criterion to define the page layout and the data sorting mode.

To define an X-axis, this parameter must be followed by the '=' character and the number of characters corresponding to the curve step (its default value is one character).

&MON : Month-by-month presentation

Used as a presentation criterion to define the page layout and the data sorting mode.

To define an X-axis, this parameter must be followed by the '=' character and the number of characters corresponding to the curve step (its default value is one character).

&YEAR : Year-by-year presentation

Used as a presentation criterion to define the page layout and the data sorting mode.

To define an X-axis, this parameter must be followed by the '=' character and the number of characters corresponding to the curve step (its default value is one character).

&SESS : Presentation by session

Used as a presentation criterion to define the page layout and the data sorting mode.

The user cannot use it to select sessions (the '=' character is therefore unnecessary).

&CHAR : Printing curve character

May only be used to define the activity calculation mode relative to the curve-type graphs.

It must follow (within parentheses) the calculation defining a curve.

&INTR : Number of input transactions

May only be used to define the activity calculation mode. Each Journal transaction is an input transaction.

&RETR : Number of real transactions

May only be used to define the activity calculation mode.

A Journal transaction is effective, provided it is not modified by another transaction and it is not itself a deletion transaction. This concept is linked to the presentation criteria, i.e. a transaction which is modified once a day is effective every day with a day-by-day presentation; it is effective only once with another presentation.

&H1 : Starting hour

This parameter is used as a Selection Criterion to define the Request Area, the Data Sorting Mode, and the Activity Calculation Mode.

This parameter has to be followed by a HHMMSS hour format.

If this parameter is missing, the starting hour coincides with the beginning of the Journal.

&H2 : End hour

This parameter is used as a Selection Criterion to define the Request Area, the Data Sorting Mode, and the Activity Calculation Mode.

This parameter has to be followed by a HHMMSS hour format.

If this parameter is missing, the end hour coincides with the beginning of the Journal.

&MIN : Minute-by-minutes presentation

Used as a presentation criterion to define the page layout and the data sorting mode.

To define an X-axis, this parameter must be followed by a '=' character and the number of characters corresponding to the curve step (its default value is one character)

&HOUR : HOUR-BY-HOURS PRESENTATION

Used as a presentation criterion to define the page layout and the data sorting mode.

To define an X-axis, this parameter must be followed by a '=' character and the number of characters corresponding to the curve step (its default value is one character)

&MCOD TRANSACTION CODE

This parameter is used as a Presentation and Selection Criterion to define the Page Layout, the Request Area, the Data Sorting Mode, and the Activity Calculation Mode.

&DSMS NUMERO D'AMELIORATION

This parameter is used as a Presentation and Selection Criterion to define the Page Layout, the Request Area, the Data Sorting Mode, and the Activity Calculation Mode.

Parameter	**AREa**	**PAGes**	**OUTput**	**OUTput**
			STA	GRA
			LIN COL	ABS ORD
&LIB	YES	YES	YES	YES
&USER	YES	YES	YES	YES
&ENTG	YES	YES	YES	YES
&ENTD		YES	YES	
&LICO	YES	YES	YES	YES
&ENT	YES	YES	YES	YES
&INPT	YES	YES	YES	YES
&D1=				
MMDDCCYY	YES		YES	YES
&D2=				
MMDDCCYY	YES		YES	YES
&S1=9999Z	YES		YES	YES
&S2=9999Z	YES		YES	YES
&MIN	YES	YES	YES	=
&HOUR	YES	YES	YES	=
&DAY	YES	YES	YES	=
&WEEK	YES	YES	YES	=
&MON	YES	YES	YES	=
&YEAR	YES	YES	YES	=
&SESS		YES	YES	
&MCOD		YES	YES	YES
&DSMS		YES	YES	YES
&CHAR				CALCULATION
&INTR				CALCULATION
&RETR				CALCULATION

= : the parameter must be followed by the separator character '=' and the curve step;

CALCULATION : only used in the Activity Calculation Mode.

ACTI: INPUTS

A '*' line with user code and password.

Specific input needed for this procedure is described in the optional utilities Reference Manual, in the chapter dedicated to this procedure.

ACTI: Description of Steps

Extraction: PTU630

Code		Type	Label
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database index
PACGGR		Input	Administration Database data
PACGGU		Input	Administration Database users
PAC7AR		Output	Development Database data
PAC7AN PAC7PJ		Output Input	Development Database index
PAC7MB		Input	Update transactions
PAC7ST		Output	Selected reports transactions (length=247)
PAC7DD		Report	Batch procedure authorization option

Return codes:

- 0: OK
- 8: Unauthorized user
- 12: System error

Printing of results: PTU640

Code		Type	Label
PAC7AE		Input	Error messages
PAC7ST		Input	Transactions for selected reports
PAC7IV		Report	Selected reports

ACTI: Execution Script

```
REM * -------------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -------------------------------------------------------
REM *              - ACTIVITY ANALYSIS -
REM *
REM * -------------------------------------------------------
REM *
REM * THE JOURNAL FILE CONTAINS ALL THE DATABASE UPDATE
```

```
REM * TRANSACTIONS. AS SUCH, IT REFLECTS USER ACTIVITY.WITH
REM * THE JOURNAL STATISTICS UTILITY (ACTI), THIS ACTIVITY
REM * CAN BE MONITORED AND PRESENTED IN THE FORM OF CHARTS.
REM * THE JOURNAL STATISTICS UTILITY ALLOWS THE DATABASE
REM * MANAGER TO QUERY THE JOURNAL BACKUP FILE BASED ON
REM * VARIOUS PARAMETERS:
REM *    - LIBRARY CODE
REM *    - USER CODE
REM *    - ENTITY TYPE
REM *    - ENTITY CODE
REM *    - LINE CODE
REM *    - TRANSACTION TYPE (C,M,D)
REM *    - DATE OF UPDATE
REM *    - SESSION NUMBER OF UPDATE
REM * ------------------------------------------------------
REM *
<job id=ACTI>

<script language="VBScript">
MyProc = "ACTI"
</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" , "PTU630"))
'--------------------------------------
WshEnv("PAC7MB") = Fic_Input
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("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7PJ") = Rep_SAVE & "\PJ"
WshEnv("PAC7ST") = Rep_TMP & "\WST.tmp"
WshEnv("PAC7DD") = Rep_USR & "\ACTIDD630.txt"
Return = WshShell.Run("BVPTU630.exe" , 1, TRUE)
If Return = 8 Then
Call Msg_Log (Array("1027"))
End If
If Return = 12 Then
Call Msg_Log (Array("1026"))
End If
Call Err_Cod(Return , 0 , "PTU630")

Call Msg_Log (Array("1022" , "PTU640"))
'-------------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7ST") = Rep_TMP & "\WST.tmp"
WshEnv("PAC7IV") = Rep_USR & "\ACTIIV640.txt"
Return = WshShell.Run("BVPTU640.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTU640")

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

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

</script>
</job>
```

Pacbench Quality Control

Introduction

The Pacbench Quality Control (PQC) facility is optional, and its use depends on the corresponding purchase agreement.

The Pacbench Quality Control facility is divided into two components:

- The Analysis component, to evaluate the quality of applications in use. This is based either on standard rules or on rules customized by the user.
- The Quality rule extraction component, customized by the user.

Two purchase options are therefore available:

- A basic option providing standard rules for quality control;
- A quality rule CUSTOMIZATION option.

The components supplied on the installation tape are:

- For both purchase options:
  - A Batch Quality Analysis procedure (PQCA);
  - A set of 'compiled' standard quality rules, in the form of a sequential file (see the Environment & Installation manual).
- For the CUSTOMIZATION option:
  - A batch procedure for the extraction and 'compilation' of the customized rules (PQCE);
  - A data element dictionary and the user entity needed for the customization of the rules, in the form of Batch transactions that the user enters in his/her own dictionary via a Batch update (UPDT). (See the Environment & Installation manual.)

Analisis

PQCA: Introduction

The PQCA procedure carries out an analysis of the quality of the applications, according to either standard rules or user-defined rules.

Characteristics

The procedure invokes a unique program (BVPACQ), which is used as a base for links to the various programs used by the procedure.

All the programs called during the procedure are therefore considered to be sub-programs of PACQ, with which they communicate via a Communication Area and special return codes.

It is functionally identical to the GPRT procedure.

The procedure is split up into 'sub-chains', identified by a 1-position code:

- D for Dictionary
- E for Dialogue Screens (OSD)
- G PACBENCH/CS Screens (OSC)
- P for Batch Language Programs (BSD)

After two general programs (BVPACA10 and BVPACA20), common to all the chains, have been executed, the sub-chains are activated, according to the generation-print requests, in the following order:

- Screens
- Programs
- Dictionary

Each sub-chain performs an extraction (followed by a printing for GCP or GCO commands).

Once these sub-chains have been activated for the extraction of the entities to be analyzed, the PTUQ20 program performs the analysis according to the rules that it has been assigned and to the analysis parameters.

Results are printed by the BVPTUQ24, PBVTUQ25 and BVPTUQ30 programs.

The processing of the generated flow in the case of generation requests is identical to that of the GPRT procedure.

Execution conditions

None. The files can remain available for on-line use.

PQCA: Inputs

See the PQC Reference Manual.

PQCA: Description of Steps

Quality analysis: PACQ

Code		Type	Label
PAC7AR		Input	Development Database data
PAC7AN		Input	Development Database index
PAC7AY		Input	Development Database extension data
PAC7AJ		Input	Development Database Journal file
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database index
PACGGR		Input	Administration Database data
PACGGU		Input	Administration Database users
PACQMF		Input	Quality rules
PAC7SC		Input	Batch generation language skeleton
PAC7SG		Input	Dialogue generation skeleton
PAC7SN		Input	Client/server meta-entity skeleton
PAC7SS		Input	Map skeleton
PAC7ME		Input	Entity inputs to be analyzed
PACQMC		Input	Selection parameter inputs
PAC7IA		Report	PACQ execution output
PAC7ID		Report	Documentation
PACQIB		Report	Selection parameter check
PACQIE		Report	Results by entity type

Code		Type	Label
PACQIF		Report	Results by entity
PACQIG		Report	List of identifiers exceeding the identificator limits
PAC7GB		Report	DBD generated
PAC7GE		Report	Dialogue generated
PAC7GG		Report	OLSD generated
PAC7GP		Report	Batch language generated
PAC7GV		Report	GDP generated
PAC7GB		Output	Concatenation of generated flows
.....			Other files mentioned in the procedure are temporary files used in the chains.

PQCA: Execution Script

```
REM * -----------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -----------------------------------------------------
REM *         - PACBENCH QUALITY CONTROL -
REM *
REM * -----------------------------------------------------
REM *
REM * THE PQCA PROCEDURE CARRIES OUT AN ANALYSIS OF THE
REM * QUALITY OF THE APPLICATIONS, ACCORDING TO EITHER
REM * STANDARD RULES OR USER-DEFINED RULES.
REM *
REM * -----------------------------------------------------
REM *
<job id=PQCA>

<script language="VBScript">
Dim MyProc
MyProc = "PQCA"
</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" , "PACQ"))
'-------------------------------------
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("PAC7EE") = Rep_TMP & "\WEE.tmp"
WshEnv("PAC7EG") = Rep_TMP & "\WEG.tmp"
WshEnv("PAC7EP") = Rep_TMP & "\WEP.tmp"
```

```
WshEnv("PAC7EV") = Rep_TMP & "\WEV.tmp"
WshEnv("PAC7GB") = Rep_USR & "\PQCAGB.txt"
WshEnv("PAC7GE") = Rep_USR & "\PQCAGE.txt"
```

Extarction of Quality Rules

PQCE: Introduction

The PQCE procedure performs the extraction of quality rules created by the user in his/her database via the user entity supplied with the Customization option of the Pacbench Quality Control Facility.

It extracts the user entity occurrences that make up the customized quality rule dictionary, checks the information, and builds a file with the 'compiled' quality rules required by the Analysis of application quality (PQCA).

For further details, see the Pacbench Quality Control Reference Manual.

Execution conditions

None. The files can remain available for on-line use.

PQCE: Inputs - Processes - Results

The user input of the PQCE procedure is similar to that of the EXUE extractor (PACX procedure).

One '*' line per library to be consulted for extraction:

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 status if Tests session
28	1	l	Language code (F=French, A=English)
29	4	EXUE	Extractor code

For further details, see Chapter 'PACX: Extractions' in this manual.

One command line:

Pos.	Len.	Value	Meaning
2	4	W1EX	Line code
6	1	Y	Identifier of UEOs extraction
7	1	U C	Library selection code: Selected library Selected library + higher level lib.
8	2	5Q	Type code of user entity dedicated to Quality Control

Result

The output of the PQCE procedure is a file containing the 'compiled' customized quality rules, which can be processed by the PQCA procedure.

PRINTED OUTPUT

This procedure prints:

- An occurrence-extraction report.
- A check report on the validity and usage of quality indicators.
- Descriptive reports on quality rules:
  - List of quality factors and criteria,
  - Description of each quality indicator,
  - Quality Control Dictionary.

PQCE: 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 inputs
PAC7BM		Input/Output	User inputs
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 printing of pregram chains
PAC7DD		Report	Printing of abends on input transactions
PAC7EE		Report	Extraction reports
PAC7EP		Report	Extraction reports
PAC7EQ		Report	Extraction reports

Code		Type	Label
PAC7EZ		Report	Extraction reports

Compilation of quality rules: PTUQ10

This step creates the customized quality rule file that will be used by the PQCA analysis procedure.

Code		Type	Label
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database index
PACGGR		Input	Administration Database data
PACGGU		Input	Administration Database users
PACGGY		Input	Administration Database Extension
PAC7AR		Input	Development Database data
PACQMI		Output	'Compiled' quality rules (length=80)
PAC7MB		Input	User inputs
PACQMC		Input	User input instances
PACQML		Output	Preparation for printing
PACQIC		Report	Rule-validity report
PAC7DD		Report	Batch procedure authorization option

Printing of quality rules: PTUQ15

Code		Type	Label
PAC7AE		Input	Error messages
PACQML		Input	Preparation for printing
PACQII		Report	List of quality factors and criteria and description by indicator
PACQIJ		Report	Dictionary of quality rules

PQCE: Execution Script

```
REM * -----------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -----------------------------------------------------
REM *          - PACBENCH QUALITY CONTROL EXTRACTION -
REM *
REM * -----------------------------------------------------
REM *
REM * FORMAT OF TRANSACTIONS AT INPUT :
REM * .. A USER AND LIBRARY LINE
REM * .. A COMMAND LINE PER ENTITY TO BE EXTRACTED
REM *    COL  2-6  : 'W1EX$'
REM *    COL  7    : SELECTION CODE OF THE LIBRARY
REM *                'U'(LIBRARY ONLY)
REM *                'C'(LIBRARY AND HIGHER LEVEL LIBRAIRIES)
REM *    COL  8-9  : TYPE CODE OF THE USER ENTITY (2 CHAR.)
REM *
REM * -----------------------------------------------------
REM *
```

```
<job id=PQCE>

<script language="VBScript">
Dim MyProc
MyProc = "PQCE"
</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("PAC7BM") = Rep_TMP & "\WBM"
WshEnv("PAC7DD") = Rep_USR & "\PQCEDD.tmp"
WshEnv("PAC7EE") = Rep_USR & "\PQCEEE.tmp"
WshEnv("PAC7EZ") = Rep_USR & "\PQCEEZ.tmp"
WshEnv("PAC7EP") = Rep_USR & "\PQCEEP.tmp"
WshEnv("PAC7EQ") = Rep_USR & "\PQCEEQ.tmp"
WshEnv("PAC7GY") = "NUL"
WshEnv("PAC7IA") = Rep_TMP & "\PQCE.IA"
WshEnv("PAC7MB") = Fic_Input
WshEnv("PAC7MJ") = "NUL"
WshEnv("PAC7MM") = Rep_TMP & "\WMM.tmp"
WshEnv("PAC7MR") = "NUL"
WshEnv("PAC7MV") = "NUL"
WshEnv("PAC7PJ") = "NUL"
WshEnv("PAC7RE") = "NUL"
WshEnv("PAC7RM") = "NUL"
WshEnv("SYSEXT") = Rep_USR & "\PQCESYS"
WshEnv("PAC7TD") = "NUL"
WshEnv("PAC7TE") = "NUL"
WshEnv("PAC7UE") = Rep_TMP & "\WUE.tmp"
WshEnv("PAC7WD") = Rep_TMP & "\WWD.tmp"
Return = WshShell.Run("BVPACX.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PACX")

Call Msg_Log (Array("1022" , "PTUQ10"))
'-------------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGY") = Rep_ABASE & "\AY"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7DD") = Rep_USR & "\PQCEDDQ10.txt"
WshEnv("PACQIC") = Rep_USR & "\PQCEICQ10.txt"
WshEnv("PACQMI") = Rep_INPUT & "MBRUL.PQC"
WshEnv("PACQMC") = Rep_TMP & "\WUE.tmp"
WshEnv("PACQML") = Rep_TMP & "\WML.tmp"
Return = WshShell.Run("BVPTUQ10.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTUQ10")

Call Msg_Log (Array("1022" , "PTUQ15"))
'-------------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PACQII") = Rep_USR & "\PQCEIIQ15.txt"
WshEnv("PAC7IJ") = Rep_USR & "\PQCEIJQ15.txt"
```

```
WshEnv("PACQML") = Rep_TMP & "\ML"
Return = WshShell.Run("BVPTUQ15.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTUQ15")

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

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


</script>
</job>
```

Chapter 6. Versioning Facilities

Standard Bridge (PCM)

Introduction

The STANDARD BRIDGE is an optional module and its use depends upon the corresponding purchase agreement. The purpose of the module is twofold:

1. Production turnover management:

This function is used to:

- Manage the generation environments, by specifying the 'production environments' which are used to manage the database freeze.
- Follow-up of entities generated from a database and managed on-site environment.
- Give the users information ralated to these entities such as their library code, the session number of the last generation, and of the last database freeze.
- Automatically freeze the database when generating into a production environment.
- To provide project follow-p to development teams in relation to generated entities.

2. Recognition of a configuration management utility

With the use of two complementary procedures, the module enable to ensure consistency between the production turnover information which are stored in the development Database, and the programs generated by a configuration management utility. But the user need to ship a file that is to be extracted from the product so as to compare it with another file extracted from the development Database.

Database Automatic Freeze

HIPM: Introduction

The purpose of the HIPM procedure is to generate transactions related to production turnover of entities and if necessary, the transactions related to the freeze of the development Database.

Execution condition

None

Abnormal endings

The procedure can be restarted once the problem has been solved.

HIPM: Inputs - Processes - Results

A '*' line with user code and password.

User input specific to the optional procedure which is used for database freeze request.

The structure of the line is as follows:

Pos.	Len.	Value	Label
2	2		Line code
		'X1'	if the entiites have been put into production
		'X4'	If no entity has been put into production
4	4	'HIST'	Freeze request
8	50		Freeze comments
58	4	ssss	Forcing of session number to be frozen: this number must be included between the current session number +1 and the current session +100

If this line is not entered, it is automatically generated when entities are put into production.

This line may be entered in order to:

- Give a specific freeze comment,
- Force the session number to be frozen.

Printed reports

This procedure prints:

- a report,
- a list of the entities used in production, and if the database has been frozen.

Results

The result of the procedure execution is a sequential file containing the production environment transactions and the optional freeze transactions.

This file must be put in input of the UPDP procedure in order to update the development Database.

HIPM: Description of Steps

Generation of production turnover transactions: PCM300

This step is used to explore the development database and generate production turnover and database freeze transactions.

Code		Type	Label
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database index
PACGGR		Input	Administration Database data
PACGGU		Input	Administration Database users
PAC7AN		Input	Development Database index
PAC7AR		Input	Development Datase data
PAC7AY		Input	Development Database random data
PAC7MB		Input	Users data .
PAC7TR		Output	Working file

Code		Type	Label
PAC7SR		Output	Working file
PAC7IG		Output	Production turnover output report
PAC7GY		Output	Production turnover transactions
PAC7DD		Report	Batch procedures authorization option

HIPM: Execution Script

```
REM * -----------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -----------------------------------------------------
REM * AUTOMATIC SESSION FREEZE
REM * -----------------------------------------------------
REM *
REM * INPUT     : USER IDENTIFICATION
REM * COL 2       '*'
REM * COL 3     : USER CODE
REM * COL 11    : PASSWORD
REM * -----------------------------------------------------
<job id=HIPM>

<script language="VBScript">
Dim MyProc
MyProc = "HIPM"
</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" , "PCM300"))
'-------------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AY") = Rep_BASE & "\AY"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7GY") = Rep_JOURNAL & "\MY"
WshEnv("PAC7DD") = Rep_USR & "\HIPMDD300.txt"
WshEnv("PAC7IG") = Rep_USR & "\HIPMIG300.txt"
WshEnv("PAC7MB") = Fic_Input
WshEnv("PAC7TR") = Rep_TMP & "\WTR.tmp"
WshEnv("PAC7SR") = Rep_TMP & "\WSR.tmp"
Return = WshShell.Run("BVPCM300.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PCM300")


Call Msg_Log (Array("1022" , "PCM039"))
'-------------------------------------
WshEnv("PAC7WY") = Rep_JOURNAL & "\WY"
WshEnv("PAC7MY") = Rep_JOURNAL & "\MY"
Return = WshShell.Run("BVPCM039.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PCM039")



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

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


</script>
</job>
```

Generation Simulation

SIPM: Introduction

The SIPM procedure is used to simulate the production turnover of entities which is usually performed by GPRT.

There are two possibilities:

- Production turnover of entities:

The information related to the entities and the environment are entered by the user.

- Transfer from one environment to another one:

The information related to the entity are provided by the source environment.

Execution condition

None.

SIPM exeution conditions are the same as GPRT's.

Abnormal execution

Whatever the reason for an abnormal ending, the procedure can be restarted as it is, once the problem has been solved.

SIPM: Inputs - Processes - Results

A '*' line with user code and pasword including information related to the procedure

Pos.	Len.	Value	Meaning
2	1	'*'	Line code
3	8	uuuuuuuu	User code
11	8	pppppppp	Password
19	3	bbb	Library code
22	4	ssss	session number (blank if current session)
26	1		Session status (' ' or 'T')
59	8	ssaammjj	Generation date , if the session is not the current session (input field for a frozen session of blank or T type, no input field for a current session)

An environment identification line 'EG' (required):

Pos.	Len.	Value	Meaning
2	2	'EG'	Line code
4	3	ttt	Type of entities processed
7	30		Target environment
37	10		Target application

Source environment line 'ES'(if transfer):

Pos.	Len.	Value	Meaning
2	2	'ES'	Line code
7	30		Source environment
37	10		Source application

Entity identification line'EU' for each entity generation to simulate

Pos.	Len.	Value	Meaning
2	2	'EU'	Line code
4	6	cccccc	Entity code
10	8	eeeeeeee	External name of the entity in target environment (if different from the database code)
18	8	nnnnnnnn	External name of the entity in source environment (if (transfer with RENAME)

Report output

This procedure prints a report

Results

Once the procedure has been executed, a sequential file is produced. This file contains the production turnover transactions.

It must be used as input to the UPDP procedure in order to update the development Database.

SIPM: Description of Steps

Work file initialization : PCMINI

Code		Type	Label
PAC7QJ		Output	Work file

Generation of simulation transactions: PCM320

Code		Type	Label
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database index
PACGGR		Input	Administration Database data

Code		Type	Label
PACGGU		Input	Administration Database users
PAC7AN		Input	Development Database index
PAC7AR		Input	Development Database data
PAC7AY		Input	Administration Database random data
PAC7MB		Input	User transaction
PAC7MT		Output	File to be used by a transfer utility
PAC7IE		Output	Simulation output report
PAC7GY		Output	Transactions for UPDP (simulation output)
PAC7DD		Report	Batch procedures authorization option

SIPM: Execution Script

```
REM * -----------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -----------------------------------------------------
REM * SIMULATION
REM * -----------------------------------------------------
REM *
REM * INPUT     : USER IDENTIFICATION
REM * COL 2       '*'
REM * COL 3     : USER CODE
REM * COL 11    : PASSWORD
REM * COL 19    : LIBRARY
REM * COL 22    : SESSION
REM * COL 25    : SESSION STATE
REM * -----------------------------------------------------
<job id=SIPM>

<script language="VBScript">
Dim MyProc
MyProc = "SIPM"
</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" , "PCM320"))
'-------------------------------------
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("PAC7GY") = Rep_JOURNAL & "\WY"
WshEnv("PAC7DD") = Rep_USR & "\SIPMDD320.txt"
WshEnv("PAC7IE") = Rep_USR & "\SIPMIE320.txt"
WshEnv("PAC7MB") = Fic_Input
WshEnv("PAC7MT") = Rep_TMP & "\WMT.tmp"
Return = WshShell.Run("BVPCM320.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PCM320")


Call Msg_Log (Array("1022" , "PCM039"))
'-------------------------------------
```

```
WshEnv("PAC7WY") = Rep_JOURNAL & "\WY"
WshEnv("PAC7MY") = Rep_JOURNAL & "\MY"
Return = WshShell.Run("BVPCM039.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PCM039")


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


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


</script>
</job>
```

Extraction of the Development Database Data

EXPM: Introduction

The EXPM procedure is used to extract from the development Database, the entities whose generation status is to be compared to the configuration management utility. The file extracted will be compared to the file extracted from the utility. The extraction will be limited to sessions, databases, environments and applications.

Execution condition

None.

Abnormal executions

The procedure can be restarted as it is once the problem has been solved.

EXPM: Inputs - Processes - Results

A '*' line with user code and password.

One or more 'S' line(s) to select the environments /application

The line is structure is as follows:

Pos.	Len.	Valeur	Meaning
2	1	'S'	Line code
3	30		Selected environment
33	10		Selected application

Report outputs

This procedure prints a report.

Results

Once the procedure has been executed, a sequential file extracted from the development Database is produced, it must be used as input to the CPPM procedure.

EXPM: Description of Steps

Extraction of the development Database: PCM200

This step explores the development Database and extracts the elements in accordance with the extraction request.

Code		Type	Label
PAC7MB		Input	User transactions
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database index
PACGGR		Input	Administration Database data
PACGGU		Input	Administration Database users
PAC7AN		Input	Development Database index
PAC7AR		Input	Development Database data
PAC7AY		Input	Development Database random data
PAC7MV		Input	User data file
PAC7ET		Output	Extraction output report
PAC7MS		Output	Extracted elements file
PAC7DD		Output	Batch procedures authorization option

Deletion of twofold extracted elements: PCM202

This step is used to suppress the elements that would be assigned by error to several extracted applications.

Code		Type	Label
PAC7AE		Input	Error messages
PAC7AN		Input	Development Database index
PAC7AR		Input	Development Database data
PAC7ME		Input	Input extracted elements file
PAC7EQ		Output	List of elements twofold extracted

Extracted elements sorting file: PCM205

This step is used to sort the extracted elements file in accordance with the criteria required to compare it with the file providing from the configuration management utility

Code		Type	Label
PAC7ME		Input	Input extracted elements file
PAC7MS		Output	Output extracted elements file

EXPM: Execution Script

```
REM * -----------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -----------------------------------------------------
REM * EXTRACTION
REM * -----------------------------------------------------
```

```
REM *
REM * INPUT     : USER IDENTIFICATION
REM * COL 2     : '*'
REM * COL 3     : USER CODE
REM * COL 11    : PASSWORD
REM * COL 19    : LIBRARY
REM * COL 22    : SESSION
REM * COL 26    : SESSION STATE
REM * ------------------------------------------------------
<job id=EXPM>

<script language="VBScript">
Dim MyProc
MyProc = "EXPM"
</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" , "PCM200"))
'-------------------------------------
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 & "\EXPMDD300.txt"
WshEnv("PAC7ET") = Rep_USR & "\EXPMET300.txt"
WshEnv("PAC7MB") = Fic_Input
WshEnv("PAC7MV") = Rep_TMP & "\WMV.tmp"
WshEnv("PAC7MS") = Rep_USR & "\EXPMMS.txt"
Return = WshShell.Run("BVPCM200.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PCM200")

Call Msg_Log (Array("1022" , "PCM202"))
'-------------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PAC7ME") = Rep_TMP & "\WMS.tmp"
WshEnv("PAC7EQ") = Rep_USR & "\EXPMEQ300.txt"
Return = WshShell.Run("BVPCM202.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PCM202")

Call Msg_Log (Array("1022" , "PCM205"))
'-------------------------------------
WshEnv("PAC7MS") = Rep_USR & "\WMS.txt"
WshEnv("PAC7ME") = Rep_TMP & "\WMS2.tmp"
Return = WshShell.Run("BVPCM205.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PCM205")


Call Msg_Log (Array("1022" , "PCM039"))
'-------------------------------------
WshEnv("PAC7WY") = Rep_JOURNAL & "\WY"
WshEnv("PAC7MY") = Rep_JOURNAL & "\MY"
Return = WshShell.Run("BVPCM039.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PCM039")


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

```
'-----------------------------------
Call DeleteFldr (Rep_TMP)


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


</script>
</job>
```

Comparison with Extracted Files

CPPM: Introduction

The CPPM procedure is used to compare a file extracted from the development Database via the EXPM procedure, with a similar file extracted by the user from the configuration management utility.

The purpose is to generate a transactions file to update the development Database via the UPDP procedure.

The transactions are due to set the development Database to the level of the configuration management utility, in relation to the production entities.

Execution conditions

The EXPM procedure must be executed at first so as to obtain a file extracted from the development Database.

Moreover, a file extracted from the configuration management utility must have been defined, with the same status as the file's extracted from the development Database

Abnormal execution

Whatever the reason for an abnormal ending, the procedure can be restarted as it is, once the problem has been solved.

CPPM: Inputs - Processes - Results

A '*' line with user code and password.

Report outputs

This procedure prints:

- A report,
- A list of entities which will be modified in the development Database after executing the UPDP procedure.

Results

Once the procedure executed, a sequential file is produced. It contains the development Database update transactions to be used as input to the UPDP procedure.

CPPM: User File

To be able to compare the development Database with the Configuration management utility used on-site, it is necessary to create a file including the data

extracted from the utility so as to compare them with the file extracted from the development Database via the EXPM procedure.

The length of the file should be 900 with the following structure:

Pos.	Len.	Value	Meaning
1	35		Parameter 1
36	35		Parameter 2
71	35		Parameter 3
106	35		Parameter 4
141	35		Parameter 5
176	35		Parameter 6
211	35		Parameter 7
246	35		Parameter 8
281	35		Parameter 9
316	35		Parameter 10
351	35		Parameter 11
386	35		Parameter 12
421	35		Parameter 13
456	35		Parameter 14
491	35		Parameter 15
526	30		Environment code
556	10		Application code
566	1		Entity type
567	6		Entity
573	8		External code of entity
585	3		Library code
588	4		Session number
592	1		Session status
593	2		Call code (when it is a user entity)
595	10		Generation data (CCYYMMDD)
605	8		Generation time
613	8		User code

The 'parameter 1' up to 'parameter 15' information, corresponds to the parameters defined in the environment screen. The sorting order is taken into account.

The 'Entity type ' and the next information corresponds to the values defined in the program generated with the name Cobol CONTANTES-PACBASE or PACBASE-CONSTANTS

CPPM: Description of Steps

Work file initialization : PCMINI

Code		Type	Label
PAC7QJ		Output	Work file

Comparison processing: PCM210

Code		Type	Label
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database index
PACGGR		Input	Administration Database data
PACGGU		Input	Administration Database users
PAC7AY		Output	Development Database extension data
PAC7AN		Input	Development Database index
PAC7AR		Input	Development Database data
PAC7MB		Input	User transactions
PAC7MP		Input	File extracted from the development Database
PAC7MU		Input	File extracted from configuration management utility
PAC7EQ		Report	Output report of check
PAC7ME		Output	File used to print comparison errors
PAC7MS		Output	File used to print update transactions
PAC7DD		Output	Batch procedures authorization option

Print of update transactions: PCM220

Code		Type	Label
PAC7AE		Input	Error messages
PAC7AN		Input	Development Database index
PAC7AR		Input	Development Database data
PAC7EQ		Report	Comparison output report
PAC7MS		Input	File used to print the results of comparison
PAC7GY		Input/Output	RRDS file containing transactions for UPDP

CPPM: Execution Script

```
REM * -----------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -----------------------------------------------------
REM * COMPARIZON
REM * -----------------------------------------------------
REM *
REM * INPUT     : USER IDENTIFICATION
REM * COL 2       '*'
REM * COL 3     : USER CODE
REM * COL 11    : PASSWORD
REM * -----------------------------------------------------
<job id=CPPM>

<script language="VBScript">
Dim MyProc
MyProc = "CPPM"
</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" , "PCM210"))
'-------------------------------------
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 & "\CPPMDD210.txt"
WshEnv("PAC7EQ") = Rep_USR & "\CPPMEQT210.txt"
WshEnv("PAC7MB") = Fic_Input
WshEnv("PAC7MP") = Rep_USR & "\EXPMMS.txt"
WshEnv("PAC7MS") = Rep_TMP & "\WMS.tmp"
WshEnv("PAC7MU") = Rep_TMP & "\WMU.tmp"
WshEnv("PAC7ME") = Rep_TMP & "\WME.tmp"
Return = WshShell.Run("BVPCM210.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PCM210")

Call Msg_Log (Array("1022" , "PCM220"))
'-------------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PAC7GY") = Rep_JOURNAL & "\WY"
WshEnv("PAC7MS") = Rep_TMP & "\WMS2.tmp"
WshEnv("PAC7EQ") = Rep_USR & "\CPPMEQ220.txt"
Return = WshShell.Run("BVPCM220.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PCM220")


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


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


</script>
</job>
```

Integrity Control of Events/Elements

CHPM: Introduction

The CHPM procedure is used to check the consistency of the whole environments and elements in the VA pac Database. The procedure produces a report printout for erroneous events and elements. The purpose of this check is to highlight the inconsistencies in the development Database at a certain point.

Execution condition

None.

Abnormal execution

The procedure can be restarted as it is, once the problem has been solved.

CHPM: Inputs - Processes - Results

Batch procedures access authorization option: A '*' line with user code and password.

Report output

This procedure prints a report listing the consistency errors detected in the development Database, and related to the environments and elements.

CHPM: Description of Steps

Environments/elements consistency check: PCM400

Code		Type	Label
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database index
PACGGR		Input	Administration Database data
PACGGU		Input	Administration Database users
PAC7AN		Input	Development Database index
PAC7AR		Input	Development Database data
PAC7AY		Input	Development Database random data
PAC7MB		Input	User transactions
PAC7MS		Output	File used to print a a check report
PAC7MV		Output	Working file
PAC7DD		Report	Batch procedures authorization option

Consistency check report printout: PCM410

Code		Type	Label
PAC7AE		Input	Error messages
PAC7EQ		Output	Output report of check
PAC7MS		Input	File used for the report printout
PAC7AN		Entrée	Index de la base de développement
PAC7AR		Entrée	Données de la Base de développement

CHPM: Execution Script

```
REM * -----------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -----------------------------------------------------
REM * VALIDATION OF THE DEVELOPMENT DATABASE
REM * -----------------------------------------------------
REM *
REM * INPUT     : USER IDENTIFICATION
REM * COL 2       '*'
REM * COL 3     : USER CODE
REM * COL 11    : PASSWORD
REM * -----------------------------------------------------
<job id=CHPM>

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

```
</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" , "PCM400"))
'-------------------------------------
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 & "\CHPMDD400.txt"
WshEnv("PAC7MB") = Fic_Input
WshEnv("PAC7MV") = Rep_TMP & "\WMV.tmp"
WshEnv("PAC7MS") = Rep_TMP & "\WMS.tmp"
Return = WshShell.Run("BVPCM400.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PCM400")

Call Msg_Log (Array("1022" , "PCM410"))
'-------------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7MS") = Rep_TMP & "\WMS.tmp"
WshEnv("PAC7EQ") = Rep_USR & "\CHPMEQ410.txt"
Return = WshShell.Run("BVPCM410.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PCM410")

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


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


</script>
</job>
```

Pac/Transfer

Introduction

The purpose of the Pac/Transfer facility is to provide an easy versioning of the developments made in a VisualAge Pacbase Database; it automates transfers of update transactions between two sessions or more.

Pac/Transfer scans the VA Pac archived Journal file and read a dedicated Parameter file.

One or more source environments are defined in this parameter file. Each can correspond with one or more target environments.

Pac/Transfer selects, from the archived Journal file, transactions that match the criteria defined via these parameters.

Pac/transfer then generates update transactions for the target environment(s) defined in the parameter file.

These transactions are used by the VA Pac batch update procedure (UPDT). If the VA Pac Database is under DSMS control, such updates are automatically included in this control.

Functionalities

Pac/Transfer is used to transfer updates made in a source session to one or several target sessions.

Once a development is completed in a test session, it is possible to transfer this session's contents onto another validation-dedicated session, and, if necessary, onto another session dedicated to production-turnover.

In the transfer file, the selected transactions from the source session are duplicated as many times as there are target sessions.

There are no constraints regarding the chronological order of sessions. It is possible to transfer the transactions entered in a given source session into a later target session (target-session number greater than that of the source session), just as it is possible to transfer it onto a previous target session (target-session number smaller than that of the source session).

The parameters transfer are stored in the administration Databse, for the whole development Databases managed by the administrator, the list of these databases is defined in the administration Database.

Thus, the développement Database notion is now important to parameterize Pac/Transfer.

You need to have defined a logical database code for each development Database.

The logical database code used is the one entered when you restore the development database with the REST procedure execution.

In batch Pac/Transfer processing procedures, it is not necessary to indicate the logical code of the development Database. The code indicated in the data file of the database processed will be systematically taken into account. This code will be used as a link between the development database and the transfer sets stored in the administration Database throughout the processing.

Processes Chronology

1. UPDATING THE TRANSFER PARAMETERS (TRUP)

Process to be executed if there are new Transaction Sets to be defined, or if parameters of existing Sets are to be modified.

2. Compressing the archived journal

Optional process (depending on the site).

3. creating the transfer file

4. preparing the DSMS environment

Process to be executed only if the Database is under DSMS control.

5. generating the transfer transactions

6. updating the development Database

7. reinitializing the DSMS environment

Process to be executed only if the Database is under DSMS control.

TRUP: Update of Transfer Parameters

TRUP: Introduction

PacTransfer processing is based on user-defined parameters stored in the UV parameters file. These parameters monitor all Pactransfer procedures.

These parameters must be created -- via a TRUP execution -- prior to any PacTransfer operation. Any change to one of these parameters must be followed by a new TRUP execution.

Several sets of transfer parameters, called Transaction Sets, may be defined. The parameter file can therefore store several Transaction Sets.

By defining several Transaction Sets, you obtain flexible transfer operations, fully adaptable to your own requirements.

Transfer parameters -- described below -- define one Transaction Set. It is not possible to set parameters common to all Sets.

Transfer parameters

- Transfer set header line:

  It is required at the beginning of the transactions relating to transfer set.

  It identifies the transfer set to which the parameters entered on the continuation lines are related.

- Session number:

  It is required to specify one source session and at least one target session.

  If you specify several target sessions, transactions entered in the source session will be transferred to each specified target session.

  NOTE: For each transfer request line, you must specify an order number so as to ensure the adequate chronology of transfers. This is particularly important when several source sessions have the same target session.

- Library:

  By default, ALL Libraries in the VisualAge Pacbase Database are taken into account for the requested source session, and the transfer target are the same Libraries.

  You may restrict the scope of a transfer by selecting one particular source Library, which then becomes the default target Library. This means that you have the wider option of selecting one or more target Libraries.

  NOTE: If the source Library is to be part of the selected target Libraries, specify its code explicitely.

If you specify several target Libraries, transactions relating to the selected source Library will be transferred to each of the target Libraries.

Example: When a transfer is defined from one source session to TWO target sessions, and from one source Library to THREE target Libraries, the volume of transferred transactions will be SIX times larger than the volume of selected transactions.

- User:

  As a default, transactions entered by ANY Database user are transferred under a unique user code.

  You may restrict the scope of the transfer by selecting one particular source user-code, which will be considered as the default target user-code. You may therefore also select a target user-code different from the selected source user-code.

- DSMS Change number:

  This type of selection refers to VisualAge Pacbase Databases under DSMS control only.

  By default, transactions associated to ANY Change are transferred under the same Change number.

  You may restrict the scope of the transfer by selecting one particular source Change-number, which will be considered as the default target Change-number. You may also select a target Change-number different from the source Change-number.

  It is also possible to transfer all transactions under a single target user-code.

  NOTE: This option overrides any target user selection such as described in Paragraph 1.3.

EXECUTION CONDITIONS

None.

Printed report

Printout of the parameter-file contents.

TRUP: Inputs

User identification line (required)

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

Transfer set header line (required)

This line must precede the update transactions of a transfer set. It identifies the transfer set to which the following transactions are related.

Pos.	Len.	Val.	Meaning
1	1		Action code
		'C'	Creation

Pos.	Len.	Val.	Meaning
		'M'	Modification
		'D'	Deletion
		' '	Creation or modification depending on the database status
		'X'	Equal to ' ', without any uppercase conversion of the label
2	2	'GA'	Line type
4	10		Transfer set code (required) different from 9999999999 and **********
14	36		Transfer set label (required in creation mode)

Session selection line

Within a Transaction Set, there must be at least one selection line of this type.

Pos.	Len.	Val.	Meaning
1	1		Action code
		'C'	Creation
		'M'	Modification
		'A'	Deletion
		' '	Creation or modification depending on the database state
		'X'	Equal to ' ', with no uppercase conversion of the label
2	2	'GS'	Line type
4	4		Source session (required)
13	2		Line number (two lines only are authorized)
		'00'	First line for the 9 first target sessions (default value)
		'01'	Continuation line for the 9 following target sessions, if required (the target session number is limited to 18: the inputs in position 1 to 7 on the first line must be repeated on the continuation line.
15	3		Sequence number of reports (required and numeric)
18	36		List of target sessions: The sessions are entered without 'T' and are not followed by blanks (one session is required at least)

Selection line of Libraries

Pos.	Len.	Val.	Meaning
1	1		Action code:
		'C'	Creation
		'M'	Modification
		'D'	Deletion
		' '	Creation or modification depending on the database state
		'X'	equal to ' ', with no uppercase conversion of the label
2	2	'GB'	Line type
4	3		Source Library (required)
13	60		Library codes (20 max.) Default: source Library Library codes are not separated by blanks.

Selection line of user codes

Pos.	Len.	Val.	Meaning
1	1		Action code
		'C'	Creation
		'M'	Modification
		'D'	Deletion
		' '	Creation or modification depending the database state
		'X'	Equal to ' ', without any uppercase conversion of the label
2	2	'GU'	Line type
4	8		Source user code (required)
13	8		Target user code Default: source user code

Selection line of DSMS changes

Pos.	Len.	Val.	Meaning
1	1		Action code:
		'C'	Creation
		'M'	Modification
		'D'	Deletion
		' '	Creation or modification depending the database state
		'X'	Equal to ' ', without any uppercase conversion of the label
2	2	'GC'	Line type
4	3		Source product code (required) it must be left-justified
7	6		Source Change number (required)
13	3		Target product code must be left-justified
16	6		Target Change number Default: Source product/Change
22	8		Target user code Default: Source user

Request line for multiple deletions

Multiple deletions may be requested at many levels:

- at the level of each type of selection
- for a given Transaction Set,
- at the level of the whole Set.

Pos.	Len.	Val.	Meaning
1	1	'B'	Multiple deletion request
2	2	'GA'	Deletion of whole Set
		'GS'	Deletion of set lines: 'GS' 'GB' GC' and 'GU'
		'GB'	Deletion of 'GB' Set line
		'GU'	Deletion of 'GU' Set line
		'GC'	Deletion of 'GC' Set line
4	10		Only if value 'GA' in columns 2 and 3.

Pos.	Len.	Val.	Meaning
		lllllllllll	Set code
		'**********'	Deletion of the whole Sets in a development Database

TRUP: Description of Steps

Update of the administration Database: BVPTUG20

This step updates the administration Database in order to store selection parameters.

Code		Type	Label
PAC7AR		Input	Development database data file
PAC7AN		Input	Development database index file
PAC7AY		Output	Development Database extension data
PACGGU		Input	Users file
PAC7AE		Input	Error messages file
PACGGR		Input/Output	Administration Database data file
PACGGN		Input/Output	Administration Database index file
PACGGY		Input/Output	Administration Database extension data file
PACGGJ		Input/Output	Administration Database Journal file
PAC7MC		Input	Parameters update transactions file
PAC7ME		Input/Output	Working file
PAC7MY		Input/Output	Working file
PAC7TB		Input/Output	Working file
PAC7BM		Output	Parameters print request file
PAC7ET		Report	Inputs check
PAC7IE		Report	Administration Database update report
PAC7IF		Report	Administration Database update errors
PAC7DD		Report	Output report: check of line '*' in relation to the development Database
PAC7DE		Report	Output report: check of line '*' in relation to the administration Database

Extraction of the administration Database: BVPTUG30

Code		Type	Label
PAC7AE		Input	Error messages
PACGGN			Administration Database index
PACGGR		Input	Administration Database data
PACGGU		Input	Administration Database users
PAC7AN		Input	Administration Database index
PAC7AR		Input	Administration Database data
PAC7AY		Input	Administration Database extension data
PAC7MB		Input	Extraction request file

Code		Type	Label
PAC7GL		Output	Target sessions list
PAC7GL		Output	Limited parameters file
PAC7DD		Report	Report on checking of line '*'
PAC7TK		Input output	Working file

Printing of selection parameters: BVPTUG31

Code		Type	Label
PAC7AE		Input	Error messages file
PAC7UY		Input	Limited parameters file
PAC7AR		Input	Administration Database data file
PAC7ET		Report	Printing of the selection parameters list

Printing of the target sessions file: BVPTUG32

Code		Type	Label
PAC7AE		Input	Error messages
PAC7AR		Input	Administration Database data
PAC7GL		Input	Target session
PAC7ET		Report	Printing of the target session list

TRUP: Execution Script

```
REM * -------------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -------------------------------------------------------
REM *        PAC/TRANSFER:
REM *        UPDATE OF THE TRANSFER PARAMETERS
REM * -------------------------------------------------------
REM *
REM * PAC/TRANSFER'S PROCESSING IS BASED ON THE USER-DEFINED
REM * PARAMETERS STORED IN THE UV PARAMETERS FILE.
REM * THESE PARAMETERS CONTROL THE VARIOUS PROCESSES OF THE
REM * FACILITY'S PROCEDURES.
REM *
REM * -------------------------------------------------------
REM *
<job id=TRUP>

<script language="VBScript">
Dim MyProc
MyProc = "TRUP"
</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" , "PTUG20"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
```

```
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PAC7AY") = "NUL"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGJ") = Rep_ABASE & "\GJ"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7BM") = Rep_TMP & "\WMB.tmp"
WshEnv("PAC7DD") = Rep_USR & "\TRUPDDG20.txt"
WshEnv("PAC7DE") = Rep_USR & "\TRUPDEG20.txt"
WshEnv("PAC7ET") = Rep_USR & "\TRUPETG20.txt"
WshEnv("PAC7IE") = Rep_USR & "\TRUPIEG20.txt"
WshEnv("PAC7IF") = Rep_USR & "\TRUPIFG20.txt"
WshEnv("PAC7MC") = Fic_Input
WshEnv("PAC7ME") = Rep_TMP & "\WME.tmp"
WshEnv("PAC7MY") = Rep_TMP & "\WMY"
WshEnv("PAC7TB") = Rep_TMP & "\WTB.tmp"
Return = WshShell.Run("BVPTUG20.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTUG20")


Call Msg_Log (Array("1022", "PTUG30"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_ABASE & "\AN"
WshEnv("PAC7AR") = Rep_ABASE & "\AR"
WshEnv("PAC7AY") = "NUL"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7DD") = Rep_USR & "\TRUPDDG30.txt"
WshEnv("PAC7GL") = Rep_TMP & "\WGL.tmp"
WshEnv("PAC7MB") = Rep_TMP & "\WMB.tmp"
WshEnv("PAC7TK") = Rep_TMP & "\WTK.tmp"
WshEnv("PAC7UY") = Rep_SAVE & "\UY"
Return = WshShell.Run("BVPTUG30.EXE" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTUG30")

Call Msg_Log (Array("1022" , "PTUG31"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AR") = Rep_ABASE & "\AR"
WshEnv("PAC7UY") = Rep_SAVE & "\UY"
WshEnv("PAC7ET") = Rep_USR & "\TRUPETG31.txt"
Return = WshShell.Run("BVPTUG31.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTUG31")


Call Msg_Log (Array("1022" , "PTUG32"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AR") = Rep_ABASE & "\AR"
WshEnv("PAC7GL") = Rep_TMP & "\WGL.tmp"
WshEnv("PAC7ET") = Rep_USR & "\TRUPETG32.txt"
Return = WshShell.Run("BVPTUG32.EXE" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTUG32")

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

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


</script>
</job>
```

Print of Transfer Parameters

TRED: Introduction

This procedure is used to print the whole transfer parameters, per development Database and transfer set.

It is possible to print the whole parameters, or to print only one development Database.

TRED: Inputs

User identification line on (required).

Pos.	Len.	Value	Meaning
2	1	'*'	Line code
3	8	uuuuuuuu	User code
11	8	pppppppp	Password
29	4		Selection of the development Database to be printed
		bbbb	Selection of a database
		'****'	Selection of all databases

TRED: Description of Steps

Checking of the process request: BVPTUG28

Code		Type	Label
PAC7AE		Input	Error messages
PACGGR		Input	Administration database data
PACGGN		Input	Administration database index
PACGGU		Input	Administration database users
PAC7AR		Input	Administration database data
PAC7MB		Input	Parameters print request
PAC7BM		Output	Parameters print request
PAC7DD		Report	Output report: user code validity check
PAC7ET		Report	Output report: print request check

Extraction of the administration Database: BVPTUG30

Code		Type	Label
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database index
PACGGR		Input	Administration Database data
PAC7GGU		Input	Administration Database users
PAC7AN		Input	Administration Database index
PAC7AR		Input	Administration Database data
PAC7AY		Input	Administration Databasea extension data
PAC7MB		Input	Printing request file
PAC7GL		Output	Target sessions file
PAC7UY		Output	Limited parameters file

Code		Type	Label
PAC7TK		Input/Output	Working file
PAC7DD		Report	Report on checking of line '*'

Printing of selection parameters: BVPTUG31

Code		Type	Label
PAC7AE		Input	Error messages
PAC7UY		Input	Limites parameters
PAC7AR		Input	Administration Database data
PAC7ET		Report	List of selection parameters

Printing of target sessions list: BVPTUG32

Code		Type	Label
PAC7AE		Input	Error messages
PAC7GL		Input	Target sessions
PAC7AR		Input	Administration Database data
PAC7ET		Report	List of target sessions

TRED: Execution Script

```
REM * -------------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -------------------------------------------------------
REM *        PAC/TRANSFER -
REM *        EDITING THE DATABASE PARAMETERS
REM * -------------------------------------------------------
REM *
REM * FOR ALL THE DATABASE OR ONE DATABASE
REM *
REM * INPUT :
REM *  -  USER IDENTIFICATION LINE (REQUIRED)
REM *
REM * -------------------------------------------------------
REM *
<job id=TRED>

<script language="VBScript">
Dim MyProc
MyProc = "TRED"
</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" , "PTUG28"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AR") = Rep_ABASE & "\AR"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
```

```
WshEnv("PAC7BM") = Rep_TMP & "\WMB.tmp"
WshEnv("PAC7DD") = Rep_USR & "\TREDDDG28.txt"
WshEnv("PAC7ET") = Rep_USR & "\TREDETG28.txt"
WshEnv("PAC7MB") = Fic_Input
Return = WshShell.Run("BVPTUG28.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTUG28")

Call Msg_Log (Array("1022", "PTUG30"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_ABASE & "\AN"
WshEnv("PAC7AR") = Rep_ABASE & "\AR"
WshEnv("PAC7AY") = "NUL"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7DD") = Rep_USR & "\TREDDDG30.txt"
WshEnv("PAC7GL") = Rep_TMP & "\WGL.tmp"
WshEnv("PAC7MB") = Rep_TMP & "\WMB.tmp"
WshEnv("PAC7TK") = Rep_TMP & "\WTK.tmp"
WshEnv("PAC7UY") = Rep_SAVE & "\UY"
Return = WshShell.Run("BVPTUG30.EXE" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTUG30")

Call Msg_Log (Array("1022" , "PTUG31"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AR") = Rep_ABASE & "\AR"
WshEnv("PAC7UY") = Rep_SAVE & "\UY"
WshEnv("PAC7ET") = Rep_USR & "\TREDETG31.txt"
Return = WshShell.Run("BVPTUG31.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTUG31")


Call Msg_Log (Array("1022" , "PTUG32"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AR") = Rep_ABASE & "\AR"
WshEnv("PAC7GL") = Rep_TMP & "\WGL.tmp"
WshEnv("PAC7ET") = Rep_USR & "\TREDETG32.txt"
Return = WshShell.Run("BVPTUG32.EXE" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTUG32")

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


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


</script>
</job>
```

TRJC: Compression of Archived Journal

TRJC: INTRODUCTION

From the VisualAge Pacbase archived Journal, the TRJC procedure produces a compressed Journal containing only useful transactions, by eliminating the intermediary transactions which are known to be useless for the transfer.

User input may include an interval of dates and/or session numbers in order to limit transfer processing to the archived Journal's transactions belonging to that interval only.

If there is no optional user input, the compression is carried out on the complete archived Journal.

You also have the possibility to erase user codes and/or Change numbers from the archived Journal. As a result, a higher rate of compression is obtained.

In this case, transfer criteria based on user codes and Changes can no longer be used.

Journal compressing is not required; it depends on the site's requirements (Journal volume, frequency of transfer operations, etc).

Execution conditions

None.

Result

A smaller archived Journal including 'useful' transactions only.

Output report

Statistical data on the TRJC execution.

TRJC: Inputs

User identification line (required)

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

Options

Pos.	Len.	Val.	Meaning
1	1		Deletion of user codes
		'0'	Yes
		'1'	No
2	1		Deletion of Change numbers
		'0'	Yes
		'1'	No
3	4		Start session number
7	4		End session number
11	8		Start date in the form CCYYMMDD
19	8		End date in the form CCYYMMDD

TRJC: Description of Steps

Compression (first stage): PTUG05

Code		Type	Label
PAC7AE		Input	Error messages
PAC7PJ		Input	Sequential image of the development Database
PAC7AN		Input	Development Database index
PACGGN		Input	Administration Database index
PACGGR		Input	Administration Database data
PACGGU PAC7MB		Input Input	Administration Database users
PAC7GP		Output	Temporary journal
PAC7ET		Report	Inputs check
PAC7DD		Report	Batch procedures error status

Compression (second phase): PTUG06

Code		Type	Label
PAC7GP		Input	Temporary Journal
PAC7PK		Output	Compressed sequential journal

Classification of deletions/creations: PTUG07

Code		Type	Label
PAC7AN		Input	Development Database index
PAC7PK		Input	Temporary journal
PAC7PL		Output	Compressed sequential journal

TRJC: Execution Script

```
REM * -----------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -----------------------------------------------------
REM *        PAC/TRANSFER -
REM *        COMPRESSION OF ARCHIVED JOURNAL
REM * -----------------------------------------------------
REM *
REM *
REM * FROM THE DATABASE ARCHIVED JOURNAL, THE TRJC
REM * PROCEDURE PRODUCES A COMPRESSED JOURNAL
REM * CONTAINING ONLY USEFUL TRANSACTIONS,
REM * BY ELIMINATING THE INTERMEDIARY TRANSACTIONS
REM * WHICH ARE KNOWN TO BE USELESS FOR THE TRANSFER.
REM *
REM * INPUT :
REM *  -  USER IDENTIFICATION LINE (REQUIRED)
REM *  -  COMMAND LINE :
REM *  COL 1  :  DELETION OF USER CODES:
REM *            '0'   YES
REM *            '1'   NO
REM *  COL 2  :  DELETION OF CHANGE NUMBERS:
REM *            '0'   YES
REM *            '1'   NO
```

```
REM *  COL 3  :  (4 CAR.) START SESSION NUMBER
REM *  COL 7  :  (4 CAR.) END SESSION NUMBER
REM *
REM *  COL 11 :  (8 CAR.) START DATE IN THE FORM CCYYMMDD
REM *  COL 19 :  (8 CAR.) END DATE IN THE FORM CCYYMMDD
REM * ------------------------------------------------------
REM *
<job id=TRJC>

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

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

<script language="VBScript">

'Au 06032001 en cours de revision par HN/GB

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

Call Msg_Log (Array("1022" , "PTUG05"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7DD") = Rep_USR & "\TRJCDDG05.txt"
WshEnv("PAC7ET") = Rep_USR & "\TRJCETG05.txt"
WshEnv("PAC7GP") = Rep_TMP & "\WGP.tmp"
WshEnv("PAC7MB") = Fic_Input
WshEnv("PAC7PJ") = Rep_BASE & "\PJ"
Return = WshShell.Run("BVPTUG05.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTUG05")

Call Msg_Log (Array("1022", "PTUG06"))
'-----------------------------------
WshEnv("PAC7GP") = Rep_TMP & "\WGP.tmp"
WshEnv("PAC7PK") = Rep_TMP & "\WPK.tmp"
Return = WshShell.Run("BVPTUG06.EXE" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTUG06")

Call Msg_Log (Array("1022" , "PTUG07"))
'-----------------------------------
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7PK") = Rep_TMP & "\WPK.tmp"
WshEnv("PAC7PL") = Rep_SAVE & "\JT"
Return = WshShell.Run("BVPTUG07.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTUG07")

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

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


</script>
</job>
```

TRPF: Creation of the Transfer File

TRPF: Introduction

From the archived Journal --whether compressed or not, depending on the site's choice and according to the contents of the Parameter file-- the TRPF procedure produces a Transfer file, which has the following characteristics:

- The only transactions processed are those meeting the source selection parameters (sessions, Libraries, users, Changes),
- The values of the selected parameters are replaced by those of the target parameters specified in the Parameter file,
- The selected transactions of the archived journal are duplicated as many times as there are target session numbers and target Library codes.

The file may contain the transactions for one, several or all of the Sets.

Execution conditions

None.

Result

The TRPF procedure produces a Transfer file, which will be used by the TRRP procedure.

TRPF: Inputs

User identification line (required)

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

Transaction Set for processing selection line (required)

Pos.	Len.	Value	Meaning
2	2	'LT'	
4	10	lllllllllll	Transaction Set for processing code
		**********	Selection of all Sets

Note:

The selection of all Sets necessarily implies that only one LT-type line be entered (with the value *****' in Positions 4 to 13).

TRPF: Description of Steps

Processing request check: BVPTUG27

Code		Type	Label
PAC7AE		Input	Error messages
PACGGR		Input	Administration Database data
PACGGN		Input	Administration Database index

Code		Type	Label
PACGGU		Input	Administration Database users
PAC7AN		Input	Development Database index
PAC7AR		Input	Development Database data
PAC7MB		Input	Parameters extraction request
PAC7BM		Output	Parameters extraction request
PAC7DD		Report	Output report: user code validity check
PAC7ET		Report	Output report: extraction request check

PAC7AR

Extraction of the administration Database: BVPTUG30

PAC7AE

Code		Type	Label
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database index
PACGGR		Input	Administration Database data
PACGGU		Input	Administration Database users
PAC7AN		Input	Administration Database index
PAC7AR		Input	Administration Database data
PAC7AY		Input	Administration Database extension data
PAC7MB		Input	EXtraction request
PAC7GL		Output	List of target sessions
PAC7UY		Output	Limited parameters for TRRP processes
PAC7DD		Report	Checking report on line '*'
PAC7TK		Output	Working file

Indexation of UY : PTUG39

Code		Type	Label
PAC7UY		Input	Limited parameters for TRRP processes
PAC7YU		Output	Limited parameters

Creation of transfers file: BVPTUG50

Code		Type	Label
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database index
PACGGR		Input	Administration Database data
PACGGU		Input	Administration Database users
PAC7AR		Input	Development Database data
PAC7JT		Input	Sequential or compressed Journal

Code		Type	Label
PAC7MB		Input	User inputs
PAC7UY		Input	Limited parameters
PAC7TJ		Output	Transfers file
PAC7DD		Report	Checking user line report
PAC7ER		Report	List of user inputs
PAC7ET		Report	Statistics on transfers

TRPF: Execution Script

```
REM * -----------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -----------------------------------------------------
REM *        PAC/TRANSFER -
REM *        CREATING THE TRANSFER FILE
REM * -----------------------------------------------------
REM *
REM * FROM THE ARCHIVED JOURNAL THE TRPF PROCEDURE PRODUCES
REM * A TRANSFER FILE.
REM *
REM * INPUT :
REM *  -  USER IDENTIFICATION LINE (REQUIRED)
REM *  -  COMMAND LINE :
REM *  COL 2  : 'LT'
REM *  COL 4  : (5 CAR.) TRANSACTION SET FOR PROCESSING CODE.
REM *           IF SELECTION OF ALL SETS '*****'
REM *
REM * -----------------------------------------------------
REM *
<job id=TRPF>

<script language="VBScript">
Dim MyProc
MyProc = "TRPF"
</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" , "PTUG27"))
'-----------------------------------
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("PAC7BM") = Rep_TMP & "\WBM.tmp"
WshEnv("PAC7DD") = Rep_USR & "\TRPFDDG27.txt"
WshEnv("PAC7ET") = Rep_USR & "\TRPFETG27.txt"
WshEnv("PAC7MB") = Fic_Input
Return = WshShell.Run("BVPTUG27.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTUG27")

Call Msg_Log (Array("1022", "PTUG30"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_ABASE & "\AN"
WshEnv("PAC7AR") = Rep_ABASE & "\AR"
```

```
WshEnv("PAC7AY") = "NUL"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7DD") = Rep_USR & "\TRPFDDG30.txt"
WshEnv("PAC7GL") = Rep_TMP & "\WGL.tmp"
WshEnv("PAC7MB") = Rep_TMP & "\WBM.tmp"
WshEnv("PAC7TK") = Rep_TMP & "\WTK.tmp"
WshEnv("PAC7UY") = Rep_SAVE & "\UY"
Return = WshShell.Run("BVPTUG30.EXE" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTUG30")

If MyUser = "" then
Rep_RI = Rep1_USR
Else
Rep_RI = Rep1_USR & "\" & MyUser
End If

Call Msg_Log (Array("1022", "PTUG39"))
'-----------------------------------
WshEnv("PAC7UY") = Rep_SAVE & "\UY"
WshEnv("PAC7YU") = Rep_RI & "\WUY.tmp"
Return = WshShell.Run("BVPTUG39.EXE" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTUG39")


Call Msg_Log (Array("1022" , "PTUG50"))
'-----------------------------------
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("PAC7DD") = Rep_USR & "\TRPFDDG50.txt"
WshEnv("PAC7ER") = Rep_USR & "\TRPFERG50.txt"
WshEnv("PAC7ET") = Rep_USR & "\TRPFETG50.txt"
WshEnv("PAC7GL") = Rep_TMP & "\WGL.tmp"
WshEnv("PAC7JT") = Rep_SAVE & "\PJ"
WshEnv("PAC7MB") = Rep_TMP & "\WBM.tmp"
WshEnv("PAC7TJ") = Rep_BASE & "\TJ"
WshEnv("PAC7UY") = Rep_RI & "\WUY.tmp"
Return = WshShell.Run("BVPTUG50.EXE" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTUG50")

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


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


</script>
</job>
```

Preparing DSMS Environment

TRDU: Introduction

The DSMS-Environment Preparation procedure (TRDU) must be used when the VisualAge Pacbase Database is under DSMS control, and when source criteria include a selected Change number.

NOTE: TRDU can operate for either one or all of the Sets defined in the Parameters file.

The VisualAge Pacbase authorizations notified for the target Change(s) must include the authorizations of the source Change(s). Otherwise, transfers in VA Pac will be rejected.

Compliance to this requirement is ensured by the TRDU procedure which temporarily aligns the target Change(s) with the source Changes regarding their VisualAge Pacbase authorizations.

NOTE: When source criteria do not include a selected Change number, TRDU cannot be applied because of the bulk of Changes involved. In this case, manual checks and alignments will be necessary.

TRDU takes into account the following additional parameters:

- If the Parameters file specifies the transfer of transactions from one source Library to one or more target Libraries, the target Change must authorize the transactions of the target Library(ies).
- If the Parameters file specifies the transfer of transactions from one source user to a target user, the target Change number must authorize the transactions under this target user code.

The TRDU procedure produces two files:

- A DSMS update-transaction file to allow target Change(s) to accept updates made on the source Change(s).

  Also, all VA Pac authorizations attached to source Changes are withdrawn. This means that during the transfer operation, no update made in VA Pac in relation to those Changes will be authorized.

  This update must be executed BEFORE the transfer operation.
- A DSMS update transactions file to set the authorizations of the source and target Changes to their initial state.

  This update must be executed AFTER the transfers are introduced in the VA Pac Database.

Execution conditions

None.

Result

Two DSMS batch update-transaction files, one of which should be applied before the transfers, the other after all transfers.

TRDU: Inputs

User identification line (required)

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

Transaction set selection line (required)

Pos.	Len.	Value	Meaning
2	2	'LT'	
4	10	llllllllll	Selected Transaction Set code
		**********	Selection of all Sets

One and only one LT-type line is required.

TRDU: Description of Steps

Check on processing request: BVPTUG26

Code		Type	Label
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database data
PACGGR		Input	Administration Database index
PAC7GU		Input	Administration Database index
PAC7AN		Input	Development Database index
PAC7AR		Input	Development Database data
PAC7MB		Input	Parameters extraction request
PAC7BM		Output	Parameters extraction request
PAC7ET		Report	Output report: check on user code validity
PAC7DD		Report	Output report: check on printing request

Extraction of the Administration Database: BVPTUG30

Code		Type	Label
PAC7AE		Input	Error messages
PACGGN		Entrée	Administration Database index
PACGGR		Input	Administration Database data
PACGGU		Input	Administration Database users
PAC7AN		Input	Administration Database index
PAC7AR		Input	Administration Database data
PAC7AY		Input	Administration Database extension data
PAC7MB		Input	Extraction request
PAC7GL		Output	Target sesssions
PAC7UY		Output	Limited parameters
PAC7DD		Report	Report outpout: checking line '*'
PAC7TK		Input/Output	Working file

Indexation of UY : PTUG39

Code		Type	Label
PAC7UY		Input	Limited parameters for TRRP processes
PAC7YU		Output	Limited parameters

Selection of sets: BVPTUG42

Code		Type	Label
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database index
PACGGR		Input	Administration Database data
PACGGU		Input	Administration Database users
PAC7AR		Input	Development Database data
PAC7UY		Input	Limited parameters
PAC7MB		Input	User inputs
PAC7BM		Output	Sets file
PAC7DD		Report	Report output: check on users
PAC7ET		Report	Report output: check on extraction

Preparation DSMS before transfer: BVPTUG44

Code		Type	Label
PAC7AE		Input	Error messages
PAC7AR		Input	Development Database data
PAC7UY		Input	Limited parameters
PACDDC		Input	Elements of the product (DSMS)
PAC7MB		Input	Batch transactions
PAC7CI		Output	Source/target initial state creation transactions
PAC7SI		Output	Source/target initial state deletion transactions
PAC7GC		Output	Target-change authorization preparation file
PAC7ET		Report	Report output

Generation of traget change transactions: BVPTUG46

Code		Type	Label
PAC7AE		Input	Error messages
PAC7AR		Input	Development Database data
PAC7GC		Input	Preparation file for target-change authorizations
PAC7CC		Output	Target before-transfer creation transactions
PAC7SC		Output	Target after-transfer deletion transactions
PAC7ET		Report	Report output

TRDU: Execution Script

```
REM * -----------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -----------------------------------------------------
REM *        PAC/TRANSFER -
REM *        PREPARING THE DSMS ENVIRONMENT
REM * -----------------------------------------------------
REM *
REM * THE DSMS-ENVIRONMENT PREPARATION PROCEDURE
```

```
REM * (TRDU) MUST BE USED WHEN THE DEVELOPMENT DATABASE
REM * IS UNDER DSMS CONTROL, AND WHEN SOURCE CRITERIA INCLUDE
REM * A SELECTED CHANGE NUMBER.
REM *
REM * INPUT :
REM *  -  USER IDENTIFICATION LINE (REQUIRED)
REM *  -  COMMAND LINE :
REM *  COL 2  : 'LT'
REM *  COL 4  : (5 CAR.) SELECTED TRANSACTION SET CODE.
REM *           IF SELECTION OF ALL SETS '*****'
REM *
REM * -------------------------------------------------------
REM *
<job id=TRDU>

<script language="VBScript">
Dim MyProc
MyProc = "TRDU"
</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" , "PTUG26"))
'-------------------------------------
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("PAC7BM") = Rep_TMP & "\WBM.tmp"
WshEnv("PAC7DD") = Rep_USR & "\TRDUDDG26.txt"
WshEnv("PAC7ET") = Rep_USR & "\TRDUETG26.txt"
WshEnv("PAC7MB") = Fic_Input
Return = WshShell.Run("BVPTUG26.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTUG26")

Call Msg_Log (Array("1022", "PTUG30"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_ABASE & "\AN"
WshEnv("PAC7AR") = Rep_ABASE & "\AR"
WshEnv("PAC7AY") = "NUL"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7DD") = Rep_USR & "\TRDUDDG30.txt"
WshEnv("PAC7GL") = Rep_TMP & "\WGL.tmp"
WshEnv("PAC7MB") = Rep_TMP & "\WBM.tmp"
WshEnv("PAC7TK") = Rep_TMP & "\WTK.tmp"
WshEnv("PAC7UY") = Rep_SAVE & "\UY"
Return = WshShell.Run("BVPTUG30.EXE" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTUG30")

Call Msg_Log (Array("1022", "PTUG39"))
'-----------------------------------
WshEnv("PAC7UY") = Rep_SAVE & "\UY"
WshEnv("PAC7YU") = Rep_TMP & "\WUY.tmp"
Return = WshShell.Run("BVPTUG39.EXE" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTUG39")


Call Msg_Log (Array("1022" , "PTUG42"))
```

```
'-----------------------------------
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("PAC7DD") = Rep_USR & "\TRDUDDG42.txt"
WshEnv("PAC7ET") = Rep_USR & "\TRDUETG42.txt"
WshEnv("PAC7BM") = Rep_TMP & "\WBM2.tmp"
WshEnv("PAC7GL") = Rep_TMP & "\WGL.tmp"
WshEnv("PAC7MB") = Rep_TMP & "\WBM.tmp"
WshEnv("PAC7UY") = Rep_TMP & "\WUY.tmp"
Return = WshShell.Run("BVPTUG42.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTUG42")


Call Msg_Log (Array("1022" , "PTUG44"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PACDDC") = Rep_BASE & "\DC"
WshEnv("PAC7MB") = Rep_TMP & "\WBM2.tmp"
WshEnv("PAC7CI") = Rep_TMP & "\WCI.tmp"
WshEnv("PAC7GC") = Rep_TMP & "\WGC.tmp"
WshEnv("PAC7SI") = Rep_TMP & "\WSI.tmp"
WshEnv("PAC7UY") = Rep_TMP & "\WUY.tmp"
WshEnv("PAC7ET") = Rep_USR & "\TRDUETG44.txt"
Return = WshShell.Run("BVPTUG44.EXE" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTUG44")


Call Msg_Log (Array("1022" , "PTUG46"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PAC7ET") = Rep_USR & "\TRDUETG46.txt"
WshEnv("PAC7CC") = Rep_TMP & "\WCC.tmp"
WshEnv("PAC7GC") = Rep_TMP & "\WGC.tmp"
WshEnv("PAC7SC") = Rep_TMP & "\WSC.tmp"
Return = WshShell.Run("BVPTUG46.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTUG46")

Call Msg_Log (Array("1022" , "COPY"))
'-----------------------------------
'COPY  SI + CC ==> TRDUAV
Call CopMFil (Rep_TMP & "\WCC.tmp" ,Rep_TMP & "\WSI.tmp" , Rep_USR & "\TRDUav.txt")
'COPY  SC + CI ==> TRDUAP
Call CopMFil (Rep_TMP & "\WSC.tmp" ,Rep_TMP & "\WCI.tmp" , Rep_USR & "\TRDUap.txt")


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


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


</script>
</job>
```

Update of DSMS before VA Pac Update

This update is performed using, as input of the DUPT procedure, the first file produced by the DSMS authorization update process.

TRRP: Generation of Transfer Transactions

TRRP: Introduction

Once the Transfer file has been built, the TRTP procedure generates transfer transactions. These have the same format as batch update transactions applicable in VA Pac by the UPDT procedure.

The transaction generation may be performed on the whole of the Transfer file or on selected parts, based on the following criteria:

1. Transaction Set (required),
2. Target Session.

Values for both criteria are indicated on the user identification line '*'. Sort options are also available and must be entered in a J-type line.

Each combination of criteria corresponds to a TRRP execution mode.

Standard execution mode (BY TRANSACTION SET):

- Transaction Set code different from '*****'
- Absence of target session

TRRP considers transactions that belong to the selected Transaction Set only. Since you have not selected a target session, transactions are generated for all target sessions found in the Parameters file regarding this Set.

However, you must run as many TRRP executions as there are target sessions:

A specific attribute -- SESSION PROCESSED -- is automatically positioned in the Parameter file once all transactions have been generated for a given session.

As a result, if this attribute is positioned for a given session (see also the other execution modes, described in Paragraphs 2 and 3), transactions for that session will not be generated and TRRP will automatically proceed with the next target session, as listed in the Parameter file.

This execution mode brings an automatic control over your transfer operations since it avoids duplicating transactions which could otherwise happen when prior TRRP executions have been run.

The TRRP standard execution mode is therefore recommended for sites where Pactransfer operations involve large volumes of transactions.

A Warning message will tell you when all sessions have been dealt with.

Generated transactions must then be used by the VisualAge Pacbase batch update procedure (UPDT).

You may prefer to concatenate all TRRP subsequent outputs and run the UPDT procedure only once.

Execution by session:

- Transaction Set code different from '*****'
- Target session: 'nnnnT' or '*****'

TRRP considers transactions that belong to the selected Transaction Set only.

- If you have selected a target session, transactions are generated for this session only.
- If you have selected all sessions ('*****'), transactions are systematically generated for all target sessions, all in one TRRP execution.

  A specific attribute -- SESSION PROCESSED -- is automatically positioned in the Parameters file once all transactions have been generated for a given session.

Generated transactions must then be used by the VA Pac batch update procedure (UPDT).

3. EXECUTION MODE FOR ALL SETS AND ALL TARGET SESSIONS:

- Transaction Set code: '*****'
- Target session number: '****'

Transactions are systematically generated for all Sets and for all their respective target sessions.

>>>>: A specific attribute -- SESSION PROCESSED -- is automatically positioned in the Parameters file once all transactions have been generated for a given session.

Generated transactions must then be used by the VA Pac batch update procedure (UPDT).

EXECUTION CONDITIONS

The Transfer file must exist (created by the TRPF procedure). Authorization level 4 is required to run a TRRP execution.

RESULT

Transfer transactions formatted for the VA Pac UPDT batch update procedure.

TRRP: Inputs

User identification line (required)

Pos.	Len.	Value	Meaning
2	1	'*'	Line code
3	8	uuuuuuuu	User code
11	8	pppppppp	Password
22	5		Selection of target session(s):
		blank	. All target sessions (default), one session processed per TRRP execution. This value cannot be used when all Transaction sets are selected
		nnnnT	. Target session number (required)
		'*****'	. All target sessions processed in one TRRP execution
33	10		Selection of Transaction Set(s):
		lllllllllll	Transaction Set code
		**********	All Transaction Sets
43	1		Formating for UPDT

Pos.	Len.	Value	Meaning
		'1'	Formating
		' '	No formating
44	1		Formating for UPDP
		'1'	Formating
		' '	No formating

Sort Options line

Pos.	Len.	Value	Meaning
2	1	'J'	Line code
4	1	' '	Chronological list
		'N'	No chronological list
5	1	' '	List by user
		'N'	No list by user
6	1	' '	List by library
		'N'	No list by library

TRRP: Description of Steps

Preparation of extraction: BVPTUG60

Code		Type	Label
PAC7AE		Input	Error messages
PACGGN		Input	Administration Database index
PACGGR		Input	Administration Database data
PACGGU		Input	Administration Database users
PAC7AR		Input	Development Databse data
PAC7JT		Input	Sequential or compressed Journal
PAC7MB		Input	User inputs
PAC7UY		Input	Limited parameters
PAC7BM		Output	Extraction request for PACX
PAC7PJ		Output	Temporary file
PAC7ET		Report	Statistics on transfer
PAC7DD		Report	Check on user

Return codes:

- 0: No error
- 8: Critical error (specified in PAC7DD)

Extraction: BVPACX

This step extracts the transcations according to user inputs

Code		Type	Label
PAC7AE		Input	Error messages

Code		Type	Label
PAC7AN		Input	Development Database index
PAC7AR		Input	Development Database data
PAC7AY		Input	Development Database extension data
PACGGN		Input	Administration Database index
PACGGR		Input	Administration Database data
PACGGU		Input	Administration Database user file
PAC7PJ		Input	Transactions selected by the Journal
PAC7MB		Input	User inputs
PAC7BM		Input/Output	User inputs
SYSEXT		Input/Output	Working file
PAC7MJ		Input/Output	Journal transactions (EXPJ)
PAC7WD		Input/Output	Extracted transactions
PAC7MV		Output	Extracted transactions for UPDT
			Extracted transactions for UPDP
PAC7IA		Report	General program-stream printout
PAC7DD		Report	Printing of errors list on input transactions
PAC7EE		Report	Extractions report output
PAC7EP		Report	Extractions report output
PAC7EQ		Report	Extractions report output
PAC7EZ		Report	Extractions report output

Return codes:

- 0: No error
- 8: Critical error (specified in PAC7DD)

Positioning the 'processing session' attribute: PTUG61

Code		Type	Label
PAC7AE		Input	Error messages
PAC7AR		Input	Development Database index
PAC7MB		Input	User inputs
PAC7UY		Input/Output	Limited parameters
PAC7ET		Report	Transfer statistics

TRRP: Execution Script

```
REM * -----------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -----------------------------------------------------
REM *        PAC/TRANSFER -
REM *        GENERATING THE TRANSFER TRANSACTIONS
REM * -----------------------------------------------------
REM *
REM * ONCE THE TRANSFER FILE HAS BEEN BUILT, THE TRRP
REM * PROCEDURE GENERATES TRANSFER TRANSACTIONS. THESE HAVE
REM * THE SAME FORMAT AS BATCH UPDATE TRANSACTIONS
```

```
REM * APPLICABLE BY THE UPDT PROCEDURE.
REM *
REM * INPUT :
REM *  -  USER IDENTIFICATION LINE (REQUIRED)
REM *     COL  2 : '*'
REM *     COL  3 : USERIDXX
REM *     COL 11 : PASSWORD
REM *     COL 22 : (5 CAR.) SELECTION OF TARGET SESSION(S)
REM *     COL 40 : (5 CAR.) SELECTION OF TRANSACTION SET(S)
REM *  -  COMMAND LINE :
REM *  COL 2  : 'J'   LINE CODE
REM *  COL 4  : ' '   CHRONOLOGICAL LIST
REM *           'N'   NO CHRONOLOGICAL LIST
REM *  COL 5  : ' '   LIST BY USER
REM *           'N'   NO LIST BY USER
REM *  COL 6  : ' '   LIST BY LIBRARY
REM *           'N'   NO LIST BY LIBRARY
REM * ------------------------------------------------------
REM *
<job id=TRRP>

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


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

<script language="VBScript">

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

If MyUser = "" then
Rep_RI = Rep1_USR
Else
Rep_RI = Rep1_USR & "\" & MyUser
End If

Call Msg_Log (Array("1022" , "PTUG60"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
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("PAC7BM") = Rep_TMP & "\WMB.tmp"
WshEnv("PAC7DD") = Rep_USR & "\TRRPDD.G60"
WshEnv("PAC7ET") = Rep_USR & "\TRRPET.G60"
WshEnv("PAC7GP") = Rep_BASE & "\GP"
WshEnv("PAC7JT") = Rep_BASE & "\TJ"
WshEnv("PAC7MB") = Fic_Input
WshEnv("PAC7PJ") = Rep_TMP & "\WPJ.tmp"
WshEnv("PAC7UY") = Rep_RI & "\WUY.tmp"
Return = WshShell.Run("BVPTUG60.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTUG60")

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("PAC7DD") = Rep_USR & "\PACXDD.txt"
WshEnv("PAC7EE") = Rep_USR & "\PACXEE.txt"
WshEnv("PAC7EP") = Rep_USR & "\PACXEP.txt"
WshEnv("PAC7EQ") = Rep_USR & "\PACXEQ.txt"
WshEnv("PAC7EZ") = Rep_USR & "\PACXEZ.txt"
WshEnv("PAC7IA") = Rep_USR & "\PACXIA.txt"
WshEnv("PAC7MV") = Rep_TMP & "\WMV.tmp"
WshEnv("PAC7BM") = Rep_TMP & "\WBM.tmp"
WshEnv("PAC7GY") = "NUL"
WshEnv("PAC7MB") = Rep_TMP & "\WMB.tmp"
WshEnv("PAC7MJ") = Rep_TMP & "\WMJ.tmp"
WshEnv("PAC7MM") = "NUL"
WshEnv("PAC7MR") = "NUL"
WshEnv("PAC7PJ") = Rep_TMP & "\WPJ.tmp"
WshEnv("PAC7RE") = "NUL"
WshEnv("PAC7RM") = "NUL"
WshEnv("PAC7TD") = "NUL"
WshEnv("PAC7TE") = "NUL"
WshEnv("PAC7UE") = "NUL"
WshEnv("PAC7WD") = Rep_TMP & "\WWD.tmp"
WshEnv("SYSEXT") = Rep_TMP & "\WSY.tmp"
Return = WshShell.Run("BVPACX.exe" , 1, TRUE)
Call Err_Cod(Return , 4 , "PACX")

Call Msg_Log (Array("1022" , "PTUG61"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AR") = Rep_BASE &  "\AR"
WshEnv("PAC7ET") = Rep_USR & "\TRRPETG61.txt"
WshEnv("PAC7MB") = Fic_Input
WshEnv("PAC7UY") = Rep_RI & "\WUY.tmp"
Return = WshShell.Run("BVPTUG60.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTUG60")


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


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


</script>
</job>
```

Update of the Development Database

The VisualAge Pacbase Database is updated via the UPDT procedure, taking the Transfer file -- created by the TRRP procedure -- as input.

In the case of a 'standard processing' of the generation of transfer transactions (see previous Subchapter), the following procedures may be executed several times:

. TRRP (Generation of transfer transactions),

. UPDT (Update of the VA Pac Database).

Reinitialization of DSMS Environment

This procedure resets update authorizations on the selected source and target Changes as they were before the transfer operation.

This initial state is obtained by running the DSMS update procedure (DUPT), using as input transactions the contents of the file resulting from the DSMS Environment Preparation procedure (TRDU).

ASCII Format

PEAS: User Parameters' ASCII format

PEAS: Description of Steps

```
ASCII SORT ON PE FILE:  PTU903

.Input backup file:
 -Original user parameters
  PAC7IN

.Output backup file:
 -User parameters sorted in ASCII sequence
  PAC7OU
```

PEAS: Command File

```
   ECHO .
   ECHO .
   ECHO *                    PEAS PROCEDURE
   ECHO *  Please note the specific parameters:
   ECHO *
   ECHO *  PE input  : complete directory and filename of PE file
   ECHO *            : %1
   ECHO *  PE output : complete directory and filename of PE file
   ECHO *            : %2
   ECHO *
   ECHO *  Example
   ECHO *  PROCPEAS C:ÈPACBASEÈSAVEÈPE.MVS C:ÈPACBASEÈSAVEÈPE
   ECHO .
   ECHO Press Control_C to stop procedure execution
   ECHO .
   REM * VA Pac : PE file ascii sort
   ECHO Execution : PTU903
   ECHO End of procedure
   ECHO Error executing PTU903
   REM * -------------------------------------------------------
   REM *      VISUALAGE PACBASE
   REM *
   REM * -------------------------------------------------------
   <job id=PEAS>

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

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

   <script language="VBScript">

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

   Dim RetRep

   If Args.Count < 3 then
    MyBox = MsgBox("PEAS. Exit : Missing files." , 48 , MyBVP)
    Wscript.Quit(1)
   End if
   If Args.Count = 3 then
    MyBox= MsgBox("PEAS. Exit : Missing output file." , 48 , MyBVP)
    Wscript.Quit(1)
```

```
End if

If FSO.FileExists(Args(2)) _
 or FSO.FileExists(Rep_TMP & "\" & Args(2)) Then
  IFile = Args(2)
Else
  RetRep = 1
  If Args(2) <> "" and Args(2) <> " " then
       AskFile = "Enter the Input File Name : " _
       & chr(10) & Args(2) & " not exist."
    Else
       AskFile = "Enter the Input File Name : "
    End If
  Do While RetRep = 1
    Reponse = InputBox(AskFile , MyBVP)
    IFile = Ucase(Reponse)
    If Not FSO.FileExists(Reponse) Then
    If Not FSO.FileExists(Rep_TMP & "\" & IFile) Then
      MyBox = MsgBox("PEAS. Exit : Input File Unknown : " _
      & Reponse  , 5 , MyBVP)
      If MyBox = 2 then
      WshVolEnv("RC") = 1
      Wscript.Quit(1)
      End If
    Else
      RetRep = 0
    End If
    Else
      RetRep = 0
    End If
  Loop
End If



  RetRep = 1
    If Args(3) <> "" and Args(3) <> " " then
       OFile = Args(3)
    Else
       AskFile = "Enter the Output File Name : "
    End If
  Do While RetRep = 1
    Reponse = InputBox(AskFile , MyBVP)
    OFile = Ucase(Reponse)
    If OFile <> "" and OFile <> " " then
      RetRep = 0
    Else
      RetRep = 1
    End If
  Loop

Call Msg_Log (Array("1022" , "ptu903"))
'-----------------------------------
WshEnv("PAC7IN") = IFile
WshEnv("PAC7OU") = OFile
Return = WshShell.Run("BVptu903.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "ptu903")

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


</script>
</job>
```

PGAS: Generation-Request Sort, ASCII Format

PGAS: Introduction

The PGAS procedure sorts the generation-request backup file (PG) as an ASCII sequence. It thus makes it possible to use this backup on ASCII platforms.

This procedure does not require any execution condition nor user input.

PGAS: Description of Steps

```
ASCII SORT ON PG FILE:  PTU906

.Input backup file:
 -Original generation requests
  PAC7IN

.Output backup file:
 -Generation requests sorted as an ASCII sequence
  PAC7OU
```

PGAS: Command File

```
   REM * -----------------------------------------------------
   REM *      VISUALAGE PACBASE
   REM *
   REM * -----------------------------------------------------
   <job id=PGAS>

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

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

   <script language="VBScript">

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

   Dim RetRep

   If Args.Count < 3 then
    MyBox = MsgBox("PGAS. Exit : Missing files." , 48 , MyBVP)
    Wscript.Quit(1)
   End if
   If Args.Count = 3 then
    MyBox= MsgBox("PGAS. Exit : Missing output file." , 48 , MyBVP)
    Wscript.Quit(1)
   End if

   If FSO.FileExists(Args(2)) _
   or FSO.FileExists(Rep_TMP & "\" & Args(2)) Then
     IFile = Args(2)
   Else
     RetRep = 1
     If Args(2) <> "" and Args(2) <> " " then
          AskFile = "Enter the Input File Name : " _
                    & chr(10) & Args(2) & " not exist."
       Else
          AskFile = "Enter the Input File Name : "
       End If
     Do While RetRep = 1
       Reponse = InputBox(AskFile , MyBVP)
       IFile = Ucase(Reponse)
       If Not FSO.FileExists(Reponse) Then
       If Not FSO.FileExists(Rep_TMP & "\" & IFile) Then
         MyBox = MsgBox("PGAS. Exit : Input File Unknown : " _
                 & Reponse  , 5 , MyBVP)
```

```
      If MyBox = 2 then
      WshVolEnv("RC") = 1
      Wscript.Quit(1)
      End If
    Else
      RetRep = 0
    End If
    Else
      RetRep = 0
    End If
  Loop
End If


  RetRep = 1
    If Args(3) <> "" and Args(3) <> " " then
       OFile = Args(3)
    Else
       AskFile = "Enter the Output File Name : "
    End If
  Do While RetRep = 1
    Reponse = InputBox(AskFile , MyBVP)
    OFile = Ucase(Reponse)
    If OFile <> "" and OFile <> " " then
      RetRep = 0
    Else
      RetRep = 1
    End If
  Loop

Call Msg_Log (Array("1022" , "PTU906"))
'-----------------------------------
WshEnv("PAC7IN") = IFile
WshEnv("PAC7OU") = OFile
Return = WshShell.Run("BVPTU906.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTU906")

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


</script>
</job>
```

PPAS: Environment Sort, ASCII Format

PPAS: Introduction

The PPAS procedure sorts the environment backup file (PP) as an ASCII sequence. It is then possible to use this backup on ASCII platforms.

This procedure does not require any execution condition nor user input.

PPAS: Description of Steps

```
ASCII SORT ON PP FILE:  PTU907

.Input backup file:
 -Original environments
  PAC7IN

.Output backup file:
 -Environments sorted as an ASCII sequence
  PAC7OU
```

PPAS: Command File

```
REM * -------------------------------------------------------
REM *      VISUALAGE PACBASE
REM *
REM * -------------------------------------------------------
<job id=PPAS>

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

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

<script language="VBScript">

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

Dim RetRep

If Args.Count < 3 then
 MyBox = MsgBox("PPAS. Exit : Missing files." , 48 , MyBVP)
 Wscript.Quit(1)
End if
If Args.Count = 3 then
 MyBox= MsgBox("PPAS. Exit : Missing output file." , 48 , MyBVP)
 Wscript.Quit(1)
End if

If FSO.FileExists(Args(2)) _
 or FSO.FileExists(Rep_TMP & "\" & Args(2)) Then
  IFile = Args(2)
Else
  RetRep = 1
  If Args(2) <> "" and Args(2) <> " " then
       AskFile = "Enter the Input File Name : " _
       & chr(10) & Args(2) & " not exist."
    Else
       AskFile = "Enter the Input File Name : "
    End If
  Do While RetRep = 1
    Reponse = InputBox(AskFile , MyBVP)
    IFile = Ucase(Reponse)
    If Not FSO.FileExists(Reponse) Then
    If Not FSO.FileExists(Rep_TMP & "\" & IFile) Then
      MyBox = MsgBox("PPAS. Exit : Input File Unknown : " _
      & Reponse  , 5 , MyBVP)
      If MyBox = 2 then
      WshVolEnv("RC") = 1
      Wscript.Quit(1)
      End If
    Else
      RetRep = 0
    End If
    Else
      RetRep = 0
    End If
  Loop
End If



  RetRep = 1
    If Args(3) <> "" and Args(3) <> " " then
       OFile = Args(3)
    Else
       AskFile = "Enter the Output File Name : "
```

```
    End If
  Do While RetRep = 1
    Reponse = InputBox(AskFile , MyBVP)
    OFile = Ucase(Reponse)
    If OFile <> "" and OFile <> " " then
      RetRep = 0
    Else
      RetRep = 1
    End If
  Loop

Call Msg_Log (Array("1022" , "PTU907"))
'-----------------------------------
WshEnv("PAC7IN") = IFile
WshEnv("PAC7OU") = OFile
Return = WshShell.Run("BVPTU907.exe" , 1, TRUE)
Call Err_Cod(Return , 0 , "PTU907")

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


</script>
</job>
```