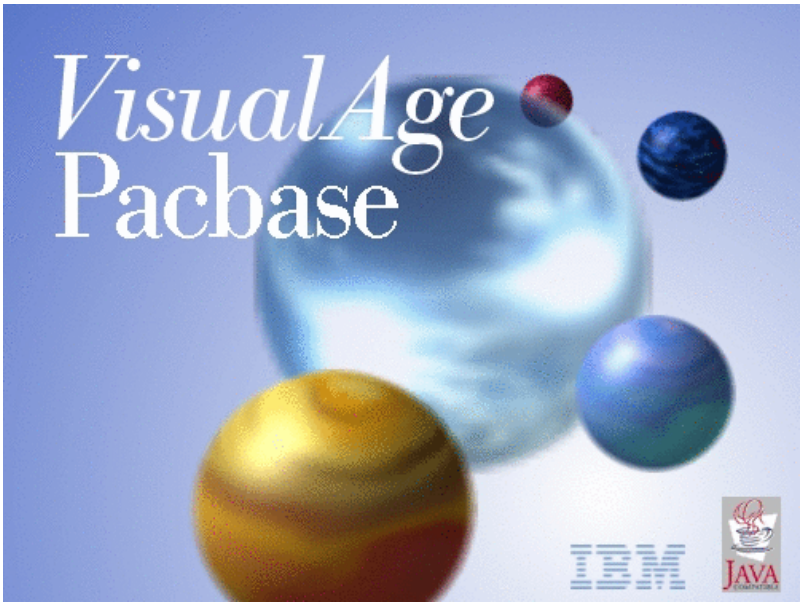
VisualAge Pacbase

The Administrator's Procedures Windows 2000 or NT Server

Version 3.5



VisualAge Pacbase

The Administrator's Procedures Windows 2000 or NT Server

Version 3.5

Note

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

You may consult or download the complete up-to-date collection of the VisualAge Pacbase documentation from the VisualAge Pacbase Support Center at:

http://www.ibm.com/support/docview.wss?rs=37&uid=swg27005477

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

Seventh Edition (May 2008)

This edition applies to the following licensed programs:
- VisualAge Pacbase Version 3.5

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

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

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

Contents

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk NY 10504–1785, U.S.A.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM France Software Laboratory - Rational Division, 1 place J.B.Clément, 93881 Noisy-Le-Grand Cedex. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

IBM may change this publication, the product described herein, or both.

Trademarks

IBM is a trademark of International Business Machines Corporation, Inc. AIX, AS/400, CICS, CICS/MVS, CICS/VSE, COBOL/2, DB2, IMS, MQSeries, OS/2, PACBASE, RACF, RS/6000, SQL/DS, TeamConnection, and VisualAge are trademarks of International Business Machines Corporation, Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and/or other countries licensed exclusively through X/Open Company Limited.

All other company, product, and service names may be trademarks of their respective owners.

Chapter 1. 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 domains:

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

A number of administration actions are carried out online in the Administrator workbench. These actions are documented in the 'AD workbench User's Guide', chapter 'Other Administration Actions'.

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.

NOTE

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

User Identification

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

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

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

Position	Length	Value	Meaning
2	1	'*'	Line code
3	8	uuuuuuuu	User code
11	8	pppppppp	Password
19	3	bbb	Library code
22	4	nnnn	Session number
26	1	'T'	Test session
27	1		With the UPDT procedure in case of multiple deletion:
		'N'	Print all transactions, including generated transactions (default option)
		'O'	Print transactions entered by the user and erroneous generated transactions
		'E'	Print erroneous transactions only
			The 2 following fields must be valued for all the extraction procedures which generate update transactions which will modify a Library/session under DSMS control (you can also indicate them on the UPDT '*' line,
40	3		Product code (3 character-code),

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

Access Authorization

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

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

Abnormal Ending

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

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

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

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

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

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

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

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

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

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

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

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

List of Run-Tile Errors

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

```
Number   Meaning
------   -------

004      Invalid file name
005      Invalid device specification
007      No more disk space
009      Directory full or does not exist
013      File not found
026      Block I-O error
027      Device not available
028      Disk space exhausted
033      Physical I-O error
105      Memory allocation error
116      Cannot allocate memory
135      File not found
150      Program abandoned on user request
157      Not enough program memory:  object file too big
         to load
170      System program not found
173      Called program file not found
188      File name too long
198      Not enough program memory: object file too large
```

```
         to load
207      Machine does not exist on the network
208      Network communication error
209      Network communication error
221 !
222 !>   Error during a SORT
223 !
```

Procedures Error Management

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

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

Advice on Use

You are strongly advised against using special characters in entity codes. The EURO character, for example, causes problems on ACU.

The Listener

The Listener must be installed in Service NT mode.

Workstations and terminals can then connect to VisualAge Pacbase.

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

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

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

When abnormal executions occur in a listener, messages can be displayed in the list of events of the 'Event Viewer' administration tool, under 'Log/Application'.

These 'ERROR'-category events can give a first indication. They generally correspond to problems in the execution environment.

However, in case of abnormal processing, the product support may ask you to activate the 'trace mode' to detect the source of the error.

- TRACE MODE

Different trace levels can be implemented:

- Level 1

  Minimum trace allowing to follow the listener processing with the calls to the COBOL communication monitor,

- Level 2

  Detailed trace of the listener processing,

- Level 4

  Trace of the messages between the listener and the client workstation.

  In the listener startup procedure, 'server.wsf', the purpose of the SRV_TRACE environment variable is to activate the trace mode. To use another trace level, you must set the SRV_TRACE variable and re-start the listener.

  EXAMPLE:

  SRV_TRACE=1 for a trace level 1

  SRV_TRACE=3 for a trace level 1 and 2

  SRV_TRACE=5 for a trace level 1 and 4

  There are two types of trace result files:

  - srv[process_number].txt

  to trace the listener (BvpServer.exe).

  - dial[process_number].txt

  to trace each listener connection (BvpDial.exe).

  These files are located in the SRV_DIR directory, set by default in server.wsf file to:

  .../data/[database_name]/tmp

  The SRV_TRACE_DEL environment variable is set to keep all or a part of the traces created by the listener execution in "dialnnn.txt" :

  SRV_TRACE_DEL : "ON" (default value)

  to keep traces produced by a processing error only.

  SRV_TRACE_DEL : "OFF"

  to keep all the traces.

3270 Emulator start-up

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

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

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

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

- code page 1147 for a French Database,

- code page 1146 for an English Database.

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

Chapter 2. Administration Database Management

ARCH - Archiving

ARCH - Introduction

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

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

The archived transactions 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 execution

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 saved.

ARCH - Input / Processing / Results

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

This procedure includes specific optional input to:

- Deactivate the previously archived transactions that are considered obsolete.
- Indicate the absence of previously archived transactions in input.
- Indicate the unavailability of the Data file (AR) in input.
- Request only the re-initialization of the transaction file.

The structure of this input is as follows:

Position	Length	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 in output.

The session number and the date are exclusive. They are ignored if the absence of input transactions is detected (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.

Warning

In this case, the transactions which were already archived are not copied to the output archived transactions file.

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

Recommendations

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 restoration 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 is lost or has been flagged as 'inconsistent', a 'D' in column 16 means that the backup procedure will not take the Data file 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 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 is lost.
- If the archived transactions file is lost or destroyed, an 'I' must be entered in column 15. As a result, the archiving procedure reformats a new sequential backup file of (archived) transactions and the previous file is lost.

You are strongly advised against indicating these options in operation JCLs. It is recommended to duplicate them in a temporary JCL before the execution.

If one of these columns is accidentally set, and if the archiving 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) files to obtain (+1) file.
- 'D' in col. 16: The archiving procedure must be re-executed BEFORE any update on the Database. If an update is performed, the Database will have to be restored completely. However the freeze transactions will be lost.
- '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 which states 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 on-line transactions 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

Archiving 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 archiving.
- updates the file of archived transactions.

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database index
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database data
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database users
PAC7JP	Save dir. : PJ	Input	Previously archived transactions
PAC7AJ	Journal dir. : AJ	Input	Journal to reinitialize from Administration Database
PAC7MB	User input	Input	User transaction
PAC7BM	Tmp dir. : WBM	Output	User transaction
PAC7AR	Database dir. : AR	Input/ Output	Development Database data
PAC7PJ	Save dir. : PJ-new	Output	Archived update transactions
PAC7PQ	NUL	Output	Deactivated transactions (length=170) : modify file name to save deactivated transactions
PAC7EU	User dir. : ARCHEU300	Report	Output report
PAC7DD	User dir. : ARCHDD300	Report	Batch procedures authorization option

Return code:

- 0: No error detected on the files.
- 4: Error in journal record (Date or session number not numeric).
- 8: No access authorization for batch procedure OR invalid Database (in this case, restart the procedure with 'D' in column 16 of the user input).
- 12: Input-output error on a file.

Re-initialization of the journal file: PTU320

This step executes two types of operations:

- Creates a record in the Journal file,
- Repositions the flag of the data file.

Code	Physical name	Type	Label
PAC7BM	Tmp dir. : WBM	Input	User transaction
PAC7AR	Database dir. : AR	Input/ Output	Administration Database data
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7AJ	Journal dir. : AJ	Output	Journal file to re-initialize
PAC7EU	User dir. : ARCHEU320	Report	Review of re-initialization

Return codes:

- 0: No error detected
- 8: the Database is invalid

  If the archiving and backup procedures are grouped into one job, the PTU320 return code can be tested in order to condition the execution of the backup procedure.

ARCH - Execution Script

```
' -------------------------------------------------------
'      VISUALAGE PACBASE
'
' -------------------------------------------------------
'             - ARCHIVAL OF THE JOURNAL -
'
' -------------------------------------------------------
'
' INPUT       : COMMAND FOR DEACTIVATION OF ARCHIVED
'               TRANSACTION
' COL 2       : "S"
' COL 3 TO  6 : SESSION NUMBER
' COL 7 TO 14 : DATE (CCYYMMDD)
' COL 15      : " " PRESENCE OF ARCHIVED TRANSACTION FILE
'             : "I" ABSENCE OF ARCHIVED TRANSACTION FILE
' COL 16      : " " PRESENCE OF DATA FILE (AR)
'             : "D" ABSENCE OF DATA FILE (AR)
' COL 17      : " " ARCHIVAL AND REINITIALIZATION
'             : "J" REINITIALIZATION WITHOUT ARCHIVAL
'
' IN THE ABSENCE OF INPUT (OR ERROR ON A COMMAND PARAM.)
' NO DEACTIVATION WILL TAKE PLACE, HOWEVER ARCHIVAL AND
' REINITIALIZATION WILL BE EXECUTED NORMALLY.
```

```
'
' TRANSACTIONS WHOSE SESSION (DATE) IS PRIOR OR EQUAL TO
' THE SESSION (DATE) INDICATED ARE NOT KEPT. THEY ARE
' RECOVERED IN THE FILE OF DEACTIVATED TRANSACTION.
'
' ------------------------------------------------------
<job id=ARCH>

<script language="VBScript">
Dim MyProc
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("1029" ))
'----------------------------
Call StateList (base, statusL)

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") = Rep_TMP & "\NULPQ.tmp"
'PAC7PQ not used
Call BvpEnv("PTU300","PAC7BM",Rep_TMP & "\WBM.tmp")
Call BvpEnv("PTU300","PAC7EU",Rep_USR & "\ARCHEU300.txt")
Call BvpEnv("PTU300","PAC7DD",Rep_USR & "\ARCHDD300.txt")
Call RunCmdLog ("BVPTU300")

 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 , 4 , "PTU300")



Call Msg_Log (Array("1022" , "PTU320"))
'-----------------------------------
```

```
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PAC7AJ") = Rep_JOURNAL & "\AJ"
Call BvpEnv("PTU320","PAC7BM",Rep_TMP & "\WBM.tmp")
Call BvpEnv("PTU320","PAC7EU",Rep_USR & "\ARCHEU320.txt")
Call RunCmdLog ("BVPTU320")

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

PACS - Introduction

The purpose of this procedure is to save the main files of the Administration Database as a sequential file.

The following files are saved:

- the data file (AR),
- the index file (AN),
- the extension data file (AY).

Execution conditions

On-line access must be prohibited.

Abnormal execution

Refer to subchapter '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 on the ADMIN Database

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

- 0: No error detected
- 8: Invalid Database

Printed report

Once the procedure is executed, a report containing the number of records saved in each file and the session number is printed.

PACS - Input / Processing / Results

Position	Length	Value	Meaning
2	1	'*'	Line code
3	8	uuuuuuuu	User code
11	8	pppppppp	Password
29	4	'SAVE'	Function code

PACS - Description of Steps

Formatting of the sequential image: PTU520

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database index
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database data
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database users
PAC7AR	Admin Database - Base dir. : AR	Input	Administration Database data
PAC7AY	Admin Database - Base dir. : AY	Input	Administration Database extension data
PAC7AN	Admin Database - Base dir. : AN	Input	Administration Database index
PAC7MB	User input	Input	Update transactions
PAC7PC	Admin Database - Save dir. : PC-new	Output	Sequential image of the Administration Database

Code	Physical name	Type	Label
PAC7RP	Tmp dir. : WRP	Output	Sequential image of data (length=153) (should be able to contain all the data)
PAC7NA	Tmp dir. : WNA	Output	Sequential image of indexes (length=59) (should be able to contain all the indexes)
PAC7NB	Tmp dir. : WNB	Output	Image of non-sorted indexes (length=59)
PAC7RY	Tmp dir. : WRY	Output	Sequential image of long data (length=1,019)
PAC7RQ	Tmp dir. : WRQ	Output	Intermediate storage (1 record, length=153)
PAC7ZK	Tmp dir. : WZK	Input / Output	Work file (length=1019)
PAC7EV	User dir. : PACSEV520	Report	User transactions list
PAC7EU	User dir. : PACSEU520	Report	Status of network before and after
PAC7EW	User dir. : PACSEW520	Report	Backup report
PAC7DD	User dir. : PACSDD520	Report	Abend output report

Return code:

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

Processing of 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 examine 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

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

```
' ------------------------------------------------------
'            - BACKUP OF THE DATABASE -
'
' ------------------------------------------------------
'
<job id=PACS>

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

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

<script language="VBScript">

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

Call ExistsSV(BVP_SvName & "")
Call ExistsSV(BVP_SvName & "I")
Call ExistsSV(BVP_SvName & "Y")
'----------------------------
If BVP_Sim <> "S" Then
Call Msg_Log (Array("1029" ))
Call StateList (base, statusL)
End If
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") = BVP_SvName & "-new"
WshEnv("PAC7PD") = BVP_SvName & "I-new"
WshEnv("PAC7PY") = BVP_SvName & "Y-new"
Call BvpEnv("PTU520","PAC7NA",Rep_TMP & "\WNA.tmp")
Call BvpEnv("PTU520","PAC7NB",Rep_TMP & "\WNB.tmp")
Call BvpEnv("PTU520","PAC7RP",Rep_TMP & "\WRP.tmp")
Call BvpEnv("PTU520","PAC7RQ",Rep_TMP & "\WRQ.tmp")
Call BvpEnv("PTU520","PAC7RY",Rep_TMP & "\WRY.tmp")
Call BvpEnv("PTU520","PAC7ZK",Rep_TMP & "\WZK.tmp")
Call BvpEnv("PTU520","PAC7EU",Rep_USR & "\PACSEU520.txt")
Call BvpEnv("PTU520","PAC7DD",Rep_USR & "\PACSDD520.txt")
```

```
Call BvpEnv("PTU520","PAC7EW",Rep_USR & "\PACSEW520.txt")
Call BvpEnv("PTU520","PAC7EV",Rep_USR & "\PACSEV520.txt")
Call RunCmdLog ("BVPTU520")
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") = BVP_SvName & "-new"
WshEnv("PAC7PD") = BVP_SvName & "I-new"
WshEnv("PAC7PY") = BVP_SvName & "Y-new"
Call BvpEnv("PTU530","PAC7NA",Rep_TMP & "\WNA.tmp")
Call BvpEnv("PTU530","PAC7NB",Rep_TMP & "\WNB.tmp")
Call BvpEnv("PTU530","PAC7RP",Rep_TMP & "\WRP.tmp")
Call BvpEnv("PTU530","PAC7RQ",Rep_TMP & "\WRQ.tmp")
Call BvpEnv("PTU530","PAC7RY",Rep_TMP & "\WRY.tmp")
Call RunCmdLog ("BVPTU530")
Call Err_Cod(Return , 0 , "PTU530")
Else
If Return = 8 Then
Call Msg_Log (Array("1032"))
End If
Call Err_Cod(Return , 4 , "PTU520")
End If

If Ret520 <> 4  then
Call Msg_Log (Array("1022" , "BACKUP"))
'-----------------------------------
Call Turnover(BVP_SvName & "")
Call Turnover(BVP_SvName & "I")
Call Turnover(BVP_SvName & "Y")
End If


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 recognizing each deletion, and sorting the data again according to the most frequent access order.

It uses the Administration Database backup file and rebuilds a sequential image. This resulting image file must then be restored via the REST procedure.

To keep the data updated since the last backup, execute before the Administration Database backup procedure.

The operating principle of this procedure is to rebuild the indexes associated with the data, using the 'image' of this data. It makes the best of the system performance features since it 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 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 Index file).

The REOR 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.

The updates performed after the constitution of the backup used upon the reorganization can be retrieved when the reorganized Database is being restored.

Abnormal execution

Refer to subchapter '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 - Input / Processing / Results

One '*' line with user code and password.

Optional user input of the procedure, in which you indicate:

- the entities to be purged,

- the users to be purged.

Pos.	Len.	Value	Meaning
2	1	'E'	Physical purge of entities
3			Entity (required)
	1	'Y'	Type
	2		Call type
6	30		Entity code to be purged (this code may be generic)

There must be one line per entity. The result can be viewed in the 'List of purged entities'.

To enter a generic code for the entity, complement it with as many '*' characters as necessary to reach 30 characters. If this code contains 30 '*', all the occurrences of the entity will be purged.

Pos.	Len.	Value	Meaning
2	1	'U'	Purge of users (9 users maximum per line)

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

Estimated 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.

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

Printed output

This procedure prints a report which lists the errors encountered during reorganization, and statistics on the contents of the Database.

It also prints reports with the statement 'Internal report'; their use is reserved for the product support team in case of problem.

Results

The output of this procedure is a reorganized sequential image of the Administration Database (where purges may have been performed). It does not contain gaps. Gaps will 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.
- The spaces allocated to the sorts should also be calculated with care.

REOR - Description of Steps

Input check: PTU2CL

This step checks user input and sets a return code if errors have been detected.

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error labels
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database index
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database data
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database users
PAC7PC	Admin Database - Save dir. : PC	Input	Sequential image of the Administration Database
PAC7MB	User Input	Input	Input work file
PAC7BM	Tmp dir. : WBM	Output	Formatted records
PAC7PU	Tmp dir. : WPU	Output	Entity purge transactions (length=44)
PAC7EE	User dir. : REOREE2CL	Report	Validation report
PAC7DD	User dir. : REORDD2CL	Report	Batch procedure authorization option

Return codes:

- 0: OK
- 4: Error on user input

- 8: No authorization for batch procedure

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	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7BM	Tmp dir. : WBM	Input	User transactions
PAC7PC	Admin Database - Save dir. : PC	Input	Administration Database sequential image
PAC7PR	Tmp dir. : WPR	Output	Formatted records (length=176 size = ND)
PAC7NX	Tmp dir. : WNX	Output	Long data
PAC7NY	Tmp dir. : WNY	Output	Bulk data
PAC7AU	Tmp dir. : WAU	Output	PR image (length=153)
PAC7PY	NUL	Output	PY image (length=1,036)
PAC7EE	User dir. : REOREE200	Report	Retrieval statistics output report

ASCII Sort: PTU205

Code	Physical name	Type	Label
PAC7PR	Tmp dir. : WPR	Input	Sorted data records
PAC7RP	Tmp dir. : WRP	Output	ASCII sorted data records

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

Purge: PTU210

This step reformats the records.

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7PR	Tmp dir. : WRP	Input	Sorted records
PAC7PU	Tmp dir. : WPU	Input	Entity records to be purged

Code	Physical name	Type	Label
PAC7BM	Tmp dir. : WBM	Input	User transactions
PAC7QS	Tmp dir. : WQS	Output	Purged records (length=176, size= ND)
PAC7UM	Tmp dir. : WUM	Output	Macro-structure call lines (length=176)
PAC7EE	User dir. : REOREE210	Report	Library and session purge report
PAC7EK	User dir. : REOREK210	Report	Entity-purge report
PAC7EB	User dir. : REOREB210	Report	Technical report

Return codes:

- 0: OK
- 8: Capacity overflow

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

Index rebuilding: PTU220

This step rebuilds indexes from the data.

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages file
PAC7BM	Tmp dir. : WBM	Input	User transactions
PAC7UR	Tmp dir. : WQS	Input	Purged data
PAC7NX	Tmp dir. : WNX	Input	Long data
PAC7UM	Tmp dir. : WUM	Input	Macro-structure call lines
PAC7PA	Tmp dir. : WPA	Output	Work file
PAC7PB	Tmp dir. : WPB	Output	Data (length=153 size=NC)
PAC7PC	Tmp dir. : WPC	Output	First data record (length=153)
PAC7AN	Tmp dir. : WAN	Output	Temporary index file (length=60 size=NI)
PAC7MR	Tmp dir. : WMR	Input/ Output	Macro-structure call lines

Code	Physical name	Type	Label
PAC7EE	User dir. : REOREE220	Report	Report of index building

ASCII Sort: PTU225

Code	Physical name	Type	Label
PAC7AN	Tmp dir.: WAN	Input	Sorted index
PAC7NA	Tmp dir.: WNA	Output	ASCII Sorted index

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

Extension data processing: PTU226

Code	Physical name	Type	Label
PAC7NY	Tmp dir. : WNY	Input	Bulk data
PAC7PA	Tmp dir. : WPA	Input	Work file
PAC7PB	Tmp dir. : WPB	Input	Data
PAC7PC	Tmp dir. : WPC	Input	First data record
PAC7QA	Tmp dir. : WQA	Output	Work file
PAC7QB	Tmp dir. : WQB	Output	Data
PAC7QC	Tmp dir. : WQC	Output	First data record (length=153)
PAC7QY	Tmp dir. : WQY	Output	Long data (length=1018)

Merge: PTU240

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

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7AN	Tmp dir. : WNA	Input	Sorted index
PAC7AU	Tmp dir. : WAU	Input	PR image
PAC7BM	Tmp dir. : WBM	Input	User transactions
PAC7PA	Tmp dir. : WQA	Input	Work file
PAC7PB	Tmp dir. : WQB	Input	Data

Code	Physical name	Type	Label
PAC7PC	Tmp dir. : WQC	Input	First data record
PAC7QY	Tmp dir. : WQY	Input	Extension data
PAC7CP	Admin Database - Save dir. : PC-new	Output	Sequential image of the Administration Database
PAC7IE	User dir. : REORIE240	Report	Building of the logical Database

Deletion of long data file: IDCAMS

This step executes a DELETE of the work file which contains the long data.

Deletion of bulk data file: IDCAMS

This step executes a DELETE of the work file which contains the bulk data.

REOR - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'                - REORGANIZATION OF THE DATABASE -
'
' ------------------------------------------------------
'
' THE REOR PROCEDURE MAY BE USED IN TWO CASES:
' . WHEN PART OF THE DATA WAS DELETED BECAUSE OF A MAL-
'  FUNCTION 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;
'
' ------------------------------------------------------
'
<job id=REOR>

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

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

<script language="VBScript">

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

Call ExistsSV(BVP_SvName & "")
```

```
Call ExistsSV(BVP_SvName & "I")
Call ExistsSV(BVP_SvName & "Y")

Call Msg_Log (Array("1022" , "PTU2CL"))
'-----------------------------------
'Example of Input File extracted from PACX/EXPU :
' Call BvpEnv("PTU2CL","PAC7MB",RepT_USR & "\PACXMR.txt")
' The first line must contain User/Password information
'With  RepT_USR is Global User Directory.

WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7MB") = Fic_Input
Call BvpEnv("PTU2CL","PAC7BM",Rep_TMP & "\WBM.tmp")
Call BvpEnv("PTU2CL","PAC7PU",Rep_TMP & "\WPU.tmp")
Call BvpEnv("PTU2CL","PAC7DD",Rep_USR & "\REORDD2CL.txt")
Call BvpEnv("PTU2CL","PAC7EE",Rep_USR & "\REOREE2CL.txt")
WshEnv("PAC7PC") = BVP_SvName & ""
Call RunCmdLog ("BVPTU2CL")
If Return = 4 Then
Call Msg_Log (Array("1057"))
End If
If Return = 8 Then
Call Msg_Log (Array("1028"))
End If
Call Err_Cod(Return , 0 , "PTU2CL")

Call Msg_Log (Array("1022" , "PTU200"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7PC") = BVP_SvName & ""
WshEnv("PAC7PY") = BVP_SvName & "Y"
Call BvpEnv("PTU200","PAC7BM",Rep_TMP & "\WBM.tmp")
Call BvpEnv("PTU200","PAC7PR",Rep_TMP & "\WPR.tmp")
Call BvpEnv("PTU200","PAC7NX",Rep_TMP & "\WNX.tmp")
Call BvpEnv("PTU200","PAC7NY",Rep_TMP & "\WNY.tmp")
Call BvpEnv("PTU200","PAC7AU",Rep_TMP & "\WAU.tmp")
Call BvpEnv("PTU200","PAC7EE",Rep_USR & "\REOREE200.txt")
Call RunCmdLog ("BVPTU200")
Call Err_Cod(Return , 0 , "PTU200")

Call Msg_Log (Array("1022" , "PTU205"))
'-----------------------------------
Call BvpEnv("PTU205","PAC7PR",Rep_TMP & "\WPR.tmp")
Call BvpEnv("PTU205","PAC7RP",Rep_TMP & "\WRP.tmp")
Call RunCmdLog ("BVPTU205")
Call Err_Cod(Return , 0 , "PTU205")
'OK :
Call DelFile (Rep_TMP & "\WPR.tmp")

Call Msg_Log (Array("1022" , "PTU210"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
```

```
Call BvpEnv("PTU210","PAC7EB",Rep_USR & "\REOREB210.txt")
Call BvpEnv("PTU210","PAC7EE",Rep_USR & "\REOREE210.txt")
Call BvpEnv("PTU210","PAC7EK",Rep_USR & "\REOREK210.txt")
Call BvpEnv("PTU210","PAC7BM",Rep_TMP & "\WBM.tmp")
Call BvpEnv("PTU210","PAC7PR",Rep_TMP & "\WRP.tmp")
Call BvpEnv("PTU210","PAC7PU",Rep_TMP & "\WPU.tmp")
Call BvpEnv("PTU210","PAC7QS",Rep_TMP & "\WQS.tmp")
Call BvpEnv("PTU210","PAC7UM",Rep_TMP & "\WUM.tmp")
Call RunCmdLog ("BVPTU210")
If Return = 8 Then
Call Msg_Log (Array("1056"))
End If
If Return = 12 Then
Call Msg_Log (Array("1026"))
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"
Call BvpEnv("PTU220","PAC7BM",Rep_TMP & "\WBM.tmp")
Call BvpEnv("PTU220","PAC7AN",Rep_TMP & "\WAN.tmp")
Call BvpEnv("PTU220","PAC7MR",Rep_TMP & "\WMR.tmp")
Call BvpEnv("PTU220","PAC7NX",Rep_TMP & "\WNX.tmp")
Call BvpEnv("PTU220","PAC7PA",Rep_TMP & "\WPA.tmp")
Call BvpEnv("PTU220","PAC7PB",Rep_TMP & "\WPB.tmp")
Call BvpEnv("PTU220","PAC7PC",Rep_TMP & "\WPC.tmp")
Call BvpEnv("PTU220","PAC7UM",Rep_TMP & "\WUM.tmp")
Call BvpEnv("PTU220","PAC7UR",Rep_TMP & "\WQS.tmp")
Call BvpEnv("PTU220","PAC7EE",Rep_USR & "\REOREE220.txt")
Call RunCmdLog ("BVPTU220")
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"))
'-----------------------------------
Call BvpEnv("PTU225","PAC7AN",Rep_TMP & "\WAN.tmp")
Call BvpEnv("PTU225","PAC7NA",Rep_TMP & "\WNA.tmp")
Call RunCmdLog ("BVPTU225")
Call Err_Cod(Return , 0 , "PTU225")

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

```
Call Msg_Log (Array("1022" , "PTU226"))
'-----------------------------------
Call BvpEnv("PTU226","PAC7PA",Rep_TMP & "\WPA.tmp")
Call BvpEnv("PTU226","PAC7PB",Rep_TMP & "\WPB.tmp")
Call BvpEnv("PTU226","PAC7PC",Rep_TMP & "\WPC.tmp")
Call BvpEnv("PTU226","PAC7QA",Rep_TMP & "\WQA.tmp")
Call BvpEnv("PTU226","PAC7QB",Rep_TMP & "\WQB.tmp")
Call BvpEnv("PTU226","PAC7QC",Rep_TMP & "\WQC.tmp")
Call BvpEnv("PTU226","PAC7QY",Rep_TMP & "\WQY.tmp")
Call BvpEnv("PTU226","PAC7NY",Rep_TMP & "\WNY.tmp")
Call RunCmdLog ("BVPTU226")
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"
Call BvpEnv("PTU240","PAC7AN",Rep_TMP & "\WNA.tmp")
Call BvpEnv("PTU240","PAC7AU",Rep_TMP & "\WAU.tmp")
Call BvpEnv("PTU240","PAC7BM",Rep_TMP & "\WBM.tmp")
Call BvpEnv("PTU240","PAC7PA",Rep_TMP & "\WQA.tmp")
Call BvpEnv("PTU240","PAC7PB",Rep_TMP & "\WQB.tmp")
Call BvpEnv("PTU240","PAC7PC",Rep_TMP & "\WQC.tmp")
Call BvpEnv("PTU240","PAC7QY",Rep_TMP & "\WQY.tmp")
WshEnv("PAC7CP") = BVP_SvName & "-new"
WshEnv("PAC7PD") = BVP_SvName & "I-new"
WshEnv("PAC7PY") = BVP_SvName & "Y-new"
Call BvpEnv("PTU240","PAC7IE",Rep_USR & "\REORIE240.txt")
Call RunCmdLog ("BVPTU240")
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(BVP_SvName & "")
Call Turnover(BVP_SvName & "I")
Call Turnover(BVP_SvName & "Y")
End if

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

</script>
</job>
```

REST - Restoration

REST - Introduction

The purpose of this procedure is to rebuild the Administration Database by using 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.

To keep users and profiles updated since the last backup, execute before the Administration Database backup procedure.

Execution conditions

The Administration Database must be closed to on-line use.

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

Abnormal execution

Refer to subchapter 'Abnormal Endings' in chapter 'Overview'.

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

REST - Input / Processing / Results

A '*' line with user code and password.

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

Position	Length	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 (EN or FR)
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 retrieval of archived transactions

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 parameter to '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.

Printed report

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.

Note: 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

User input recognition: PTU010

Code	Physical name	Type	Label
CARTE	User input	Input	User parameters
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database index
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database data
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database users

Code	Physical name	Type	Label
PAC7PC	Save dir. : PC	Input	Sequential image of the Administration Database
PAC7MB	Tmp dir. : WMB	Output	Parameter
PAC7DD	User dir. : RESTDD010	Report	Batch procedure authorization option

Return code:

- 8: No authorization on batch procedures

Validation of journal contents: PTU380

This step is executed if the Journal file exists.

Code	Physical name	Type	Label
PAC7MB	Tmp dir. : WMB	Input	User transactions
PAC7AE	System - Skel dir. : AE	Input	Error messages file
PAC7AJ	Journal dir. : AJ	Input	Journal file
PAC7EU	User dir. : RESTEU380	Report	(only if the Journal was not archived)

Return code:

- 0 : The Journal file was archived
- 8 : The Journal file was not archived (none of the REST steps was executed).

Restoration of the Database: PTU400

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

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error labels
PAC7MB	Tmp dir. : WMB	Input	User transactions
PAC7PC	Admin Database - Save dir. : PC	Input	Sequential image of the Administration Database
PAC7AR	Admin Database - Base dir. : AR	Output	Administration Database data

Code	Physical name	Type	Label
PAC7AY	Admin Database - Base dir. : AY	Output	Administration Database extension data
PAC7AN	Admin Database - Base dir. : AN	Output	Administration Database index
PAC7AJ	Journal dir. : AJ	Output	Administration Database Journal
PAC7PS	Tmp dir. : WPS	Output	Work file (2 records, length=144)
PAC7EU	User dir. : RESTEU400	Report	Restoration report
PAC7DD	User dir. : RESTDD400	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.

Warning: this step is REQUIRED to obtain a consistent Database.

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7MB	Tmp dir. : WMB	Input	User transactions
PAC7AR	Admin Database - Base dir. : AR	Input/ Output	Administration Database data
PAC7JO	Save dir. : PJ	Input	Journal to apply
PAC7PS	Tmp dir. : WPS	Input	Work file
PAC7OJ	Tmp dir. : WOJ	Output	Update transactions (length=170)
PAC7EU	User dir. : RESTEU420	Report	Retrieval report

Return codes:

- 0: There are transactions to be retrieved
- 4: No transaction to be retrieved

  OR erroneous user input.

In case of abnormal ending, the Database cannot be updated.

Update of the Administration Database: PACA15

Code	Physical name	Type	Label
PAC7AR	Admin Database - Base dir. : AR	Output	Administration Database data
PAC7AN	Admin Database - Base dir. : AN	Output	Administration Database index
PAC7AY	Admin Database - Base dir. : AY	Output	Administration Database extension
PAC7AJ	Journal dir. : AJ	Output	Administration Database Journal
PAC7AE	System - Skel. dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database index
PACGGR	Admin. Database - Base dir. : AR	Input	Administration Database data
PACGGY	Admin. Database - Base dir. : AY	Input	Administration Database Extension
PACGGU	Admin. Database - Base dir. : GU	Input	Administration Database users
PAC7DC	Base dir. : DC	Input	DSMS file of the Development Database
PAC7ME	NUL	Input	Work file
PAC7MV	Tmp dir. : WOJ	Input	Update transactions
PAC7RB	NUL	Output	UPDT erroneous transactions (length=80)
PAC7RY	NUL	Output	UPDP erroneous transactions (length=310)
PAC7IE	User dir. : RESTIEA15	Report	Update report (length=132)
PAC7IF	User dir. : RESTIFA15	Report	Erroneous transactions list (length=132)

The list of a user's transactions is preceded by a banner specifying the user's code.

Return codes :

- 0: OK, no error
- 2: warning
- 4: error

REST - Execution Script

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

Call Msg_Log (Array("1029" ))
'---------------------------
Call StateList (base, statusL)

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

Dim Ret420
```

```
Ret420 = 0

' No Rollback
WshEnv("BVPRB") = "N"

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"
Call BvpEnv("PTU010","PAC7DD",Rep_USR & "\RESTDD010.txt")
Call BvpEnv("PTU010","PAC7MB",Rep_TMP & "\WMB.tmp")
WshEnv("PAC7PC") = BVP_SvName & ""
WshEnv("PAC7PD") = BVP_SvName & "I"
WshEnv("PAC7PY") = BVP_SvName & "Y"
Call RunCmdLog ("BVPTU010")
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"
Call BvpEnv("PTU380","PAC7EU",Rep_USR & "\RESTEU380.txt")
Call BvpEnv("PTU380","PAC7MB",Rep_TMP & "\WMB.tmp")
Call RunCmdLog ("BVPTU380")
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"
Call BvpEnv("PTU400","PAC7EU",Rep_USR & "\RESTEU400.txt")
Call BvpEnv("PTU400","PAC7MB",Rep_TMP & "\WMB.tmp")
WshEnv("PAC7PC") = BVP_SvName & ""
WshEnv("PAC7PD") = BVP_SvName & "I"
Call BvpEnv("PTU400","PAC7PS",Rep_TMP & "\WPS.tmp")
WshEnv("PAC7PY") = BVP_SvName & "Y"
Call RunCmdLog ("BVPTU400")
WshVolEnv("RC") = Return
```

```
Call Err_Cod(Return , 0 , "PTU400")

If Not FSO.FileExists( Rep_SAVE & "\PJ"  ) Then
        Call Msg_Log (Array("1004", "PJ in SAVE"))
        Return = 1
        WshVolEnv("RC") = Return
        Wscript.Quit (Return)
End If


Call Msg_Log (Array("1022" , "PTU420"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AR") =  Rep_BASE & "\AR"
WshEnv("PAC7JO") =  Rep_SAVE & "\PJ"
Call BvpEnv("PTU420","PAC7MB",Rep_TMP & "\WMB.tmp")
Call BvpEnv("PTU420","PAC7OJ",Rep_TMP & "\WOJ.tmp")
Call BvpEnv("PTU420","PAC7PS",Rep_TMP & "\WPS.tmp")
Call BvpEnv("PTU420","PAC7EU",Rep_USR & "\RESTEU420.txt")
Call RunCmdLog ("BVPTU420")
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"
Call BvpEnv("PACA15","PAC7IE",Rep_USR & "\RESTIEA15.txt")
Call BvpEnv("PACA15","PAC7IF",Rep_USR & "\RESTIFA15.txt")
WshEnv("SEMLOCK") = Rep_BASE & "\LO"
WshEnv("SEMADMIN") = Rep_ABASE & "\LO"
Call BvpEnv("PACA15","PAC7MV",Rep_TMP & "\WOJ.tmp")
WshEnv("PAC7ME") = Rep_TMP & "\NUL.tmp"
'PAC7ME not used, on default
```

```
WshEnv("PAC7RB") = Rep_TMP & "\NULRB.tmp"
'PAC7RB not used, on default
WshEnv("PAC7RY") = Rep_TMP & "\NULRY.tmp"
'PAC7RY not used, on default
Call RunCmdLog ("BVPACA15")
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


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

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

</script>
</job>
```

PAGX - Extractions

PAGX - Introduction

The extraction procedure allows to perform extractions from the Administration Database via a PAF extractor (selection criteria).

The extracted data are formatted as transactions and therefore can be used in input to the UPGP procedure.

Execution conditions

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

PAGX - User Input

One user line.

Position	Length	Value	Meaning
2	1	'*'	Line code
3	8	uuuuuuuu	User code
11	8	pppppppp	Password
19	3	'***'	Extraction library code

Position	Length	Value	Meaning
29	4	'EXTR'	Extractor code
34	1	'1'	Formatting for UPGP (PAF)
50	1	' '	No transfer of password
		'1'	Password transfer

One or two command lines per entity to be extracted.

Position	Length	Value	Meaning
2	1	'W'	Line code
3	1	'1'	Line number
4	2	'EX'	
6	1	'C'	Library selection code:
7	3		Entity
		'YAB'	VAP Database
		'YAF'	Command line
		'YAR'	Pac/Transfer
		'YAT'	Parameter
		'YAU'	User
		'YAV'	VAP Profile
		'YD1'	Publishing's Document
		'YEN'	Endevor Type
10	30		Code of the Entity

Printed output

The procedure produces a sorted list of extracted entities.

PAGX - Description of Steps

Extraction: PAGX

This step extracts the transactions according to user input.

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7AN	Admin Database - Base dir. : AN	Input	Administration Database index

Code	Physical name	Type	Label
PAC7AR	Admin Database - Base dir. : AR	Input	Administration Database data
PAC7AY	Admin Database - Base dir. : AY	Input	Administration Database extension data
PACGGN	Admin Dataase - Base dir. : AN	Input	Administration Database index
PACGGR	Admin. Database - Base dir. : AR	Input	Administration Database data
PACGGU	Admin. Database - Base dir. : GU	Input	Administration Database user file
PAC7PJ	Save dir. : PJ	Input	Transactions selected in the Journal
PAC7MB	User Input	Input	User input
PAC7BM	Tmp dir. : WBM	Input/Output	User input
PAC7MM	NUL		
SYSEXT	Tmp dir. : WSY	Input/Output	Work file
PAC7MJ	NUL		
PAC7TE	NUL		
PAC7RE	NUL		
PAC7RM	NUL		
PAC7WD	Tmp dir. : WWD	Input/Output	Extracted transactions
PAC7MV	NUL		
PAC7MR	NUL		
PAC7TD	NUL		
PAC7MX	NUL		
PAC7GY	User dir. : PAGXGY	Output	Extracted transactions for UPGP
PAC7UE	NUL		
PAC7IA	User dir. : PAGXIA	Report	General program-stream printout
PAC7DD	User dir. : PAGXDD	Report	Printing of errors on input transactions
PAC7ED	User dir. : PAGXED	Report	Extractions report
PAC7EE	User dir. : PAGXEE	Report	Extractions report
PAC7EG	User dir. : PAGXEG	Report	Extractions report
PAC7EM	User dir. : PAGXEM	Report	Extractions report

Code	Physical name	Type	Label
PAC7EP	User dir. : PAGXEP	Report	Extractions report
PAC7EQ	User dir. : PAGXEQ	Report	Extractions report
PAC7ES	NUL		
PAC7EU	User dir. : PAGXEU	Report	Extractions report
PAC7EZ	User dir. : PAGXEZ	Report	Extractions report
PAC7MA	NUL		

Return codes:

- 0: No error
- 4: Error on user input (detailed in PAC7EE) or on the extraction (detailed in PAC7EZ)
- 8: Error on '*' line (detailed in PAC7DD)

PAGX - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'           - EXTRACTIONS FROM DATABASE -
' ------------------------------------------------------
'
' THE PAGX PROCEDURE ALLOWS TO PERFORM DATA EXTRACTIONS
' FROM THE ADMINISTRATION DATABASE VIA PAF EXTRACTOR.
'
' ------------------------------------------------------
'
<job id=PAGX>

<script language="VBScript">
Dim MyProc
MyProc = "PAGX"
</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" , "PAGX"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_ABASE & "\AN"
WshEnv("PAC7AR") = Rep_ABASE & "\AR"
WshEnv("PAC7AY") = Rep_ABASE & "\AY"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
```

```
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7PJ") = Rep_ASAVE & "\PJ"
WshEnv("PAC7MB") = Fic_Input
Call BvpEnv("PACX","PAC7BM",Rep_TMP & "\WBM.tmp")
Call BvpEnv("PACX","PAC7WD",Rep_TMP & "\WWD.tmp")
WshEnv("PAC7MM") =  Rep_TMP & "\NUL.tmp"
'PAC7MM not used, on default
WshEnv("PAC7MJ") =  Rep_TMP & "\NULMJ.tmp"
'PAC7MJ not used, on default
WshEnv("PAC7TE") =  Rep_TMP & "\NULTE.tmp"
'PAC7TE not used, on default
WshEnv("PAC7RE") =  Rep_TMP & "\NULRE.tmp"
'PAC7RE not used, on default
WshEnv("PAC7RM") =  Rep_TMP & "\NULRM.tmp"
'PAC7RM not used, on default
WshEnv("PAC7MA") = Rep_TMP & "\NUL.tmp"
'PAC7MA not used, on default
WshEnv("PAC7ES") = Rep_TMP & "\NUL.tmp"
'PAC7ES not used, on default

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

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

WshEnv("PAC7UE") =  Rep_TMP & "\NULUE.tmp"
'PAC7UE not used, on default

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

WshEnv("PAC7TD") =  Rep_TMP & "\NULTD.tmp"
'PAC7TD not used, on default

WshEnv("PAC7MV") =  Rep_TMP & "\NULMV.tmp"
'PAC7MV not used, on default

WshEnv("PAC7MR") =  Rep_TMP & "\NULMR.tmp"
'PAC7MR not used, on default

WshEnv("PAC7MX") =  Rep_TMP & "\NULMX.tmp"
'PAC7MX not used, on default

Call BvpEnv("PACX","PAC7IA",Rep_USR & "\PAGXIA.txt")
Call BvpEnv("PACX","PAC7DD",Rep_USR & "\PAGXDD.txt")
Call BvpEnv("PACX","PAC7ED",Rep_USR & "\PAGXED.txt")
Call BvpEnv("PACX","PAC7EE",Rep_USR & "\PAGXEE.txt")
Call BvpEnv("PACX","PAC7EG",Rep_USR & "\PAGXEG.txt")
Call BvpEnv("PACX","PAC7EM",Rep_USR & "\PAGXEM.txt")
Call BvpEnv("PACX","PAC7EP",Rep_USR & "\PAGXEP.txt")
Call BvpEnv("PACX","PAC7EQ",Rep_USR & "\PAGXEQ.txt")
Call BvpEnv("PACX","PAC7EU",Rep_USR & "\PAGXEU.txt")
Call BvpEnv("PACX","PAC7EZ",Rep_USR & "\PAGXEZ.txt")
```

```
Call BvpEnv("PACX","SYSEXT",Rep_TMP & "\WSY.tmp")
Call RunCmdLog ("BVPACX")
Call Err_Cod(Return , 4 , "PAGX")

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

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

</script>
</job>
```

UPGP - PAF Updates

UPGP - Introduction

The UPGP procedure performs an update of the Administration Database from a sequential file reflecting PAF tables.

Execution conditions

The Administration Database must be closed to on-line use in order to be updated, except for hardware environments that support concurrent on-line and batch access.

Abnormal execution

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

There are two types of abnormal executions:

- Abnormal execution which occurs before the execution of the BVPACA15 program, or during the opening of files in this program. The procedure can be restarted after the problem has been corrected.
- Abnormal execution which occurs during the execution of the BVPACA15 program. The Database is left in an inconsistent state. If the problem appeared during input-output on a Database file, the printed error message and the file status will indicate the solution.

In all cases, the procedure can be restarted only by reloading the Backup file and applying the archived transactions chived subsequent to this backup (REST procedure).

UPGP - User Input

The sequential file of input transactions is produced by the PAGX procedure. Its records are images of the PAF tables. Refer to the Pacbase Access Facility Tables documentation for a detailed description of these tables.

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

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

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

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

Position	Length	Meaning
1	1	Transaction code (C, M, X, A or D, B, S)
2	10	Table code
12	1	Record continuation code: blank character for the first record, any character for the continuation records.
13	1	Not used
14	55	Explicit keywords
69	237	Field containing columns specific to the associated Meta Entity

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

Position	Length	Meaning
1	1	Transaction code (C, M, X, A or D, B)
2	10	Table code
12	1	Record continuation code: blank character for the first record, any character for the continuation records
13	1	Not used
14	30	User Entity code

Position	Length	Meaning
44	262	Field which contains columns specific to the associated Meta Entity

Update rules

Update transactions are not sorted.

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

Position	Length	Value	Meaning
2	10	'ASSIGN'	Table code
12	8	uuuuuuuu	User code
20	8	pppppppp	Password
28	3	'***'	Library code
40	3	ppp	Product code (in case of a Database under DSMS control)
43	6	nnnnnn	Product number (in case of a Database under DSMS control)

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

(Refer to the description of the UPDT output.)

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

Printed output

The two printed output generated by this procedure are:

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

Result

Output of the UPGP procedure is a Database ready to be used on-line or in batch mode.

UPGP - Description of Steps

Transactions formatting: PAF900

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

Update of the Administration Database: PACA15

Code	Physical name	Type	Label
PAC7AR	Admin. Database - Base dir.: AR	Output	Administration Database Data file
PAC7AN	Admin. Database - Base dir.: AN	Output	Administration Database Index file
PAC7AY	Admin. Database - Base dir.: AY	Output	Administration Database extension

Code	Physical name	Type	Label
PAC7AJ	Admin. Database - Base dir.: AJ	Output	Administration Database journal
PAC7AE	System - Skel. dir.: AE	Input	Error messages
PACGGN	Admin. Database - Base dir.: AN	Input	Administration Database Index file
PACGGR	Admin. Database - Base dir.: AR	Input	Administration Database Data file
PACGGY	Admin. Database - Base dir.: AY	Input	Administration Database Extension
PACGGU	Admin. Database - Base dir.: GU	Input	Administration Database users
PAC7DC	NUL		
PAC7ME	Tmp dir.: WME	Input	Work file
PAC7MV	Tmp dir.: WMV	Input	Update transactions
PAC7RB	User dir.: RBA15	Output	UPDT erroneous transactions (length=80)
PAC7RY	User dir.: RYA15	Output	UPDP erroneous transactions (length=310)
PAC7IE	User dir.: IEA15	Report	Update report (length=132)
PAC7IF	User dir.: IFA15	Report	List of erroneous transactions (length=132)

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

Return codes:

- 0 : OK without error
- 2 : warning
- 4 : critical error

UPGP - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'            - BATCH UPDATE FROM PAF TABLES -
'
' ------------------------------------------------------
'
' THE UPGP PROCEDURE PERFORMS AN UPDATE OF THE
```

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

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

</script>

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

<script language="VBScript">

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

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


Call Msg_Log (Array("1022" , "PAF900"))
'-----------------------------------
WshEnv("PAC7GY") = Fic_Input
If Not FSO.FileExists(WshEnv("PAC7GY")) Then
  Call Msg_Log (Array("1004" , "PAC7GY"))
  ' Call Msg_Log (Array("1020"))
  Msg = Nls_Lib
  EndJob (1)
  Wscript.Quit (1)
End If

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

```
Call Err_Cod(Return , 0 , "PAF900")

Call Msg_Log (Array("1022" , "PACA15"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AJ") = Rep_AJOURNAL & "\AJ"
WshEnv("PAC7AN") = Rep_ABASE & "\AN"
WshEnv("PAC7AR") = Rep_ABASE & "\AR"
WshEnv("PAC7AY") = Rep_ABASE & "\AY"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGY") = Rep_ABASE & "\AY"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7DC") = Rep_TMP & "\NULDC.tmp"
'PAC7DC not used, on default
Call BvpEnv("PACA15","PAC7IE",Rep_USR & "\UPGPIEA15.txt")
Call BvpEnv("PACA15","PAC7IF",Rep_USR & "\UPGPIFA15.txt")
WshEnv("SEMLOCK") = Rep_BASE & "\LO"
WshEnv("SEMADMIN") = Rep_ABASE & "\LO"
Call BvpEnv("PACA15","PAC7ME",Rep_TMP & "\WME.tmp")
Call BvpEnv("PACA15","PAC7MV",Rep_TMP & "\WMV.tmp")
Call BvpEnv("PACA15","PAC7RB",Rep_TMP & "\WRB.tmp")
Call BvpEnv("PACA15","PAC7RY",Rep_USR & "\UPGPRYA15.txt")
Call RunCmdLog ("BVPACA15")
If Return = 2 Then
Call Msg_Log (Array("1061"))
End If
If Return = 4 Then
Call Msg_Log (Array("1060"))
End If
Call Err_Cod(Return , 4 , "PACA15")

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

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


</script>
</job>
```

Chapter 3. Development Databases Management

PACS - Backup Procedures

PACS - 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 - Input Common to Managers

Position	Length	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)
33	1		Library extraction code (reserved to SASN)
		'A'	Extraction of a lower-level library and of its higher-level library
		'D'	Extraction of a library and of its dependent libraries
34	1		Library extraction option (reserved to SASN)
		'I'	Extraction of inter-library also (only if col 33 = 'D')
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

Position	Length	Value	Meaning
67	1		(UXSR-reserved, not taken into account if col 26 = 'T')
		'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

  Important

  This function is part of the 'LCU Partitioned Database Manager' optional utility. Its use is therefore subject to a separate purchase agreement.
- UXSR: sub-networks extraction

There is one launch script (BVPxxxx.wsf) for each of these values. An environment variable, named WshEnv("BVP_SaveName"), is defined for each script.

Database Management

MLIB - Introduction:

The Database Management procedure has a two-fold purpose:

- Initialize the Database in the form of a sequential file PC (or three files PC, PD, PY, if the Dispatch option is used), which is (are) 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 use and, except for a simulation, the procedure must be followed by the REST procedure so that the new library structure is taken into account.

Abnormal execution

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

MLIB - Input / Processing / Results:

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 (optional) lines as there are Libraries to create, modify or delete.

The structure of the header is the following:

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

An update simulation is used to obtain the state of the Database as it would be if the requested modifications had actually been carried out.

It enables you to assess the impact of a change in the structure of the Database before its actual execution. For large Databases, actual execution may use considerable machine time.

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

Position	Length	Value	Meaning
1	1	'C'	Creation
		'M'	Modification
		'A'	Deletion
		'I'	Clear the Library's content
2	1	'*'	Line code
3	3	bbb	Code of the Library to update
6	3	ccc	Code of the upper level Library
9	1	' '	Non 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 the deletion of the entire content of this Library.. Its content is replaced by empty records, or 'gaps' (see the REST restoration procedure).

Clear Library's content transaction

Apply the same rules as for the deletion transaction.

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; on that occasion, their code can be modified.
- The Libraries which are not initialized are visible to all users, but are not ready to receive development specifications. You change their status to Initialized in the Administrator workbench.

Warning

A Development Database cannot contain more than 595 Libraries.

Modification transactions

Generally, transactions modify existing links between two Libraries. Usually a new Library is inserted between two existing Libraries.

When a new upper Library is assigned to a Library, the new Library must be empty and linked directly or indirectly to the former upper Library.

Structural errors are automatically detected.

It is not possible to cancel and recreate a Library during the same MLIB 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 after the update is printed.

Results

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

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 the recovery of disk space when Libraries are deleted. Records are physically present in the Database as 'gaps'. The Reorganization (REOR) procedure 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) saves the Database main files as sequential files: 'PC', 'PD' and 'PY' (logical names).

The backup is performed on the following files:

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

An option allows the backup of data files, indexes and extension data in three sequential files: ('AR' saved as 'PC', 'AN' saved as 'PD' and 'AY' as 'PY'). Otherwise, all three files are saved as one 'PC' file.

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 execution

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

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.

Execution of archiving and save procedures

If the save procedure is preceded by a Journal archiving (ARCH procedure), its execution may be conditioned by the return code of the PTU320 program of the ARCH procedure.

- 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.

SAVE - Input / Processing / Results:

Output

Depending on the 'Dispatch' option taken during restoration, the output of the backup procedure is the following:

- Either a single sequential file ('PC'), of variable length, containing the mirror of the three saved files,
- Or three sequential files ('PC', 'PD' and 'PY') of variable length.

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 more 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.

Note

The MLIB procedure can produce a result similar to that produced by SASN. However, it keeps data 'gaps' in the backup and, unlike the SASN procedure, does not allow physical space to be gained.

Execution conditions

The Database must be closed to on-line use.

Abnormal execution

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

SASN - User Input:

One line per library to be extracted.

Position	Length	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 AAA.

Caution

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 Input:

There are two types of specific user input:

- Heading line (required) at the top of the input file that specifies if a simulation of the Database is desired or not.
- As many lines (optional) as there are libraries to be created, modified or cancelled.

The structure of the header is the following:

Position	Length	Value	Meaning
2	1	'G'	Line code
4	1	' '	Actual update
		'S'	Simulation

The simulation allows to obtain the state of the network after modifications, without actually carrying out these modifications. You can then assess the impact of a modification in the network structure before carrying it out (it might prove to be costly in terms of machine time if the Database is very large).

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

Position	Length	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
9	3	ddd	Code of source Library required even when creating a new Library ; in this case enter any code which does not exist 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	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database index
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database data

Code	Physical name	Type	Label
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database users
PAC7AR	Database dir. : AR	Input	Development Database data
PAC7AY	Database dir. : AY	Input	Development Database extension data
PAC7AN	Database dir. : AN	Input	Development Database index
PAC7MB	User input	Input	Update transactions
PAC7PC	Save dir. : PC-new	Output	Database sequential image (length=1,023)
PAC7PD	Save dir. : PCI-new	Output	If Dispatch option of backup: sequential image 2 of the Database (length=1,023)
PAC7PY	Save dir. : PCY-new	Output	If Dispatch option of backup: sequential image 3 of the Database (length=1,023)
PAC7RP	Tmp dir. : WRP	Output	Sequential image of data (length=153) (should be able to contain all data)
PAC7NA	Tmp dir. : WNA	Output	Sequential image of index (length=59) (should be able to contain all data)
PAC7NB	Tmp dir. : WNB	Output	Image of unsorted index (length=59)
PAC7RY	Tmp dir. : WRY	Output	Sequential image of long data (length=1,019)
PAC7RQ	Tmp dir. : WRQ	Output	Temporary storage (1 record, length=153)
PAC7ZK	Tmp dir. : WZK	Input/Output	Work file (length=1019)
PAC7EV	User dir. : PACSEV520	Report	User transactions list
PAC7EU	User dir. : PACSEU520	Report	State of library before and after
PAC7EW	User dir. : PACSEW520	Report	Backup report
PAC7DD	User dir. : PACSDD520	Report	Errors report

Return codes:

- 2: MLIB or SASN and no error. Execution of PTU530.

- 4: MLIB and Database simulation
- 8: Inconsistency of the Database or no authorization on batch procedure

Formatting sequential images: PTU530

Code	Physical name	Type	Label
PAC7AR	Database dir. : AR	Input	Development Database data
PAC7RP	Tmp dir. : WRP	Input	Data sequential image
PAC7NA	Tmp dir. : WNA	Input	Index sequential image
PAC7NB	Tmp dir. : WNB	Input	unsorted index image
PAC7RY	Tmp dir. : WRY	Input	extension data sequential image
PAC7RQ	Tmp dir. : WRQ	Input	Intermediary storage
PAC7PC	Save dir. : PC-new	Output	Development Database sequential image
PAC7PD	Save dir. : PCI-new	Output	If backup Dispatch option: sequential image 2 of the network
PAC7PY	Save dir. : PCY-new	Output	If backup Dispatch option: sequential image 3 of the network

Response to return code:

If the program sends a return code greater than '2', the 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, the user should first contact the product support to analyze the problem, 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 REOR procedure.

PACS - Execution Script

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

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

```
</script>

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

<script language="VBScript">

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

Call ExistsSV(BVP_SvName & "")
Call ExistsSV(BVP_SvName & "I")
Call ExistsSV(BVP_SvName & "Y")
'----------------------------
If BVP_Sim <> "S" Then
Call Msg_Log (Array("1029" ))
Call StateList (base, statusL)
End If
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") = BVP_SvName & "-new"
WshEnv("PAC7PD") = BVP_SvName & "I-new"
WshEnv("PAC7PY") = BVP_SvName & "Y-new"
Call BvpEnv("PTU520","PAC7NA",Rep_TMP & "\WNA.tmp")
Call BvpEnv("PTU520","PAC7NB",Rep_TMP & "\WNB.tmp")
Call BvpEnv("PTU520","PAC7RP",Rep_TMP & "\WRP.tmp")
Call BvpEnv("PTU520","PAC7RQ",Rep_TMP & "\WRQ.tmp")
Call BvpEnv("PTU520","PAC7RY",Rep_TMP & "\WRY.tmp")
Call BvpEnv("PTU520","PAC7ZK",Rep_TMP & "\WZK.tmp")
Call BvpEnv("PTU520","PAC7EU",Rep_USR & "\PACSEU520.txt")
Call BvpEnv("PTU520","PAC7DD",Rep_USR & "\PACSDD520.txt")
Call BvpEnv("PTU520","PAC7EW",Rep_USR & "\PACSEW520.txt")
Call BvpEnv("PTU520","PAC7EV",Rep_USR & "\PACSEV520.txt")
Call RunCmdLog ("BVPTU520")
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") = BVP_SvName & "-new"
WshEnv("PAC7PD") = BVP_SvName & "I-new"
WshEnv("PAC7PY") = BVP_SvName & "Y-new"
Call BvpEnv("PTU530","PAC7NA",Rep_TMP & "\WNA.tmp")
Call BvpEnv("PTU530","PAC7NB",Rep_TMP & "\WNB.tmp")
Call BvpEnv("PTU530","PAC7RP",Rep_TMP & "\WRP.tmp")
Call BvpEnv("PTU530","PAC7RQ",Rep_TMP & "\WRQ.tmp")
Call BvpEnv("PTU530","PAC7RY",Rep_TMP & "\WRY.tmp")
Call RunCmdLog ("BVPTU530")
Call Err_Cod(Return , 0 , "PTU530")
Else
If Return = 8 Then
Call Msg_Log (Array("1032"))
End If
Call Err_Cod(Return , 4 , "PTU520")
End If

If Ret520 <> 4  then
Call Msg_Log (Array("1022" , "BACKUP"))
'-----------------------------------
Call Turnover(BVP_SvName & "")
Call Turnover(BVP_SvName & "I")
Call Turnover(BVP_SvName & "Y")
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) freezes the Database and updates the Database in batch mode.

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 also executed in on-line mode.

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

UPDT - Input

A '*' line with the user code and password.

Position	Length	Value	Meaning
2	6	'X1HIST'	Line code for a session freeze
8	50		Session Name
58	4		Session number
65	1		Session status
		' '	Frozen and read-write session
		'N'	Frozen and read-only session
		'D'	Deleted session
66	15		Session Short Name: optional, set to the session number by default. It must be unique, whether it is specified or not.

UPDT - Description of Steps

Transactions formatting: PACA05

Code	Physical name	Type	Label
PAC7AR	Database dir.: AR	Input	Development Database Data file
PAC7AN	Database dir.: AN	Input	Development Database Index file
PAC7AY	Database dir.: AY	Input	Development Database extension data
PAC7AE	System - Skel. dir.: AE	Input	Error messages
PACGGN	Admin. Database - Base dir.: AN	Input	Administration Database Index file
PACGGR	Admin. Database - Base dir.: AR	Input	Administration Database Data file
PACGGU	Admin. Database - Base dir.: GU	Input	Administration Database Users
PAC7MB	User input	Input	Update transactions
PAC7ME	Tmp dir.: WME	Output	Work file (length=372)

Code	Physical name	Type	Label
PAC7MV	Tmp dir. : WMV	Output	Formatted transactions (length=170, must be able to contain all input transactions plus the elementary delete transactions generated by the multiple delete transactions)
PAC7MW	Tmp dir. : WMW	Output	Work file

Update of the Development Database: PACA15

Code	Physical name	Type	Label
PAC7AR	Database dir.: AR	Output	Development Database Data file
PAC7AN	Database dir.: AN	Output	Development Database index
PAC7AY	Database dir.: AY	Output	Development Database extension
PAC7AJ	Journal dir.: AJ	Output	Development Database journal
PAC7AE	System - Skel. dir.: AE	Input	Error messages
PACGGN	Admin. Database - Base dir.: AN	Input	Administration Database Index file
PACGGR	Admin. Database - Base dir.: AR	Input	Administration Database Data file
PACGGY	Admin. Database - Base dir.: AY	Input	Administration Database Extension
PACGGU	Admin. Database - Base dir.: GU	Input	Administration Database users
PAC7DC	Base dir.: DC	Input	DSMS file of Development Database Elements
PAC7ME	Tmp dir.: WME	Input	Work file
PAC7MV	Tmp dir.: WMV	Input	Update transactions
PAC7RB	User dir. : RBA15	Output	UPDT erroneous transactions (length=80)
PAC7RY	User dir. : RYA15	Output	UPDP erroneous transactions (length=310)
PAC7IE	User dir. : IEA15	Report	Update report (length=132)
PAC7IF	User dir. : IFA15	Report	List of erroneous transactions (length=132)

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

Return codes:

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

UPDT - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'                 - BATCH UPDATE -
'
' ------------------------------------------------------
'
' REFER TO THE BATCH FORMS AND TO THE DESCRIPTION OF THE
' INPUT CORRESPONDING TO EACH ENTITY.
'
' INPUT :
'  -  USER IDENTIFICATION LINE (REQUIRED)
'     COL  2 : "*"
'     COL  3 : USERIDXX
'     COL 11 : PASSWORD
'     COL 28 : LANGUAGE CODE, USEFUL WHEN TRANSACTION ARE
'              NOT IN THE SAME LANGUAGE AS THE DATABASE.
'     COL 67 : "N"  DEFAULT VALUE WITH EXTRACTORS
'  -  COMMAND LINE
'     THE LIST OF ALL AVAILABLE VALUES FOR THE ENTITY
'     TO BE UPDATED IS FOUND IN REFERENCE MANUAL.
'
' ------------------------------------------------------
'
<job id=UPDT>

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

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

<script language="VBScript">

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

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

```
End If
Call Err_Cod(Return , 0 , "PTUBAS")

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

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

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

Call Err_Cod(Return , 0 , "PACA05")

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

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

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

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


</script>
</job>
```

SASY - Database System Backup Complement

SASY - Introduction

This Database Backup procedure saves 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 saved:

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

Execution conditions

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

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 execution

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 - User Input

A '*' line with the user code and password.

SASY - Description of Steps

Session number incrementation: PTU502

Code	Physical name	Type	Label
PAC7AR	Database dir. : AR	Input/ Output	Development Database data
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database index
PACGGR	Admin Datavase - Base dir. : AR	Input	Administration Database data
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database users
PAC7MB	User input	Input	User transactions
PAC7GZ	User dir. : SASYGZ502	Report	Report
PAC7DD	User dir. : SASYDD502	Report	Batch procedure authorization option
PAC7DS	User dir. : SASYDS502	Report	Database validity report

SASY - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'     - COMPLEMENT OF "SYSTEM" BACKUP OF THE DATABASE-
'
' ------------------------------------------------------
'
'  THE DATABASE SYSTEM BACKUP COMPLEMENT PROCEDURE
'  (SASY) ALLOWS YOU 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).
```

```
'
' ------------------------------------------------------
'
<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"
Call BvpEnv("PTU502","PAC7DD",Rep_USR & "\SASYDD502.txt")
Call BvpEnv("PTU502","PAC7DS",Rep_USR & "\SASYDS502.txt")
Call BvpEnv("PTU502","PAC7GZ",Rep_USR & "\SASYGZ502.txt")
Call RunCmdLog ("BVPTU502")
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 managed on-line, using the sequential image produced by the Backup, the Database Management (PACS), the Reorganization (REOR).

It also allows to retrieve archived transactions after this sequential image has been produced.

Execution conditions

The Database must be closed to on-line processing.

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

Abnormal execution

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

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

REST - Input / Processing / Results

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

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

Position	Length	Value	Meaning
2	1	'Y'	Line code
3	5	nnnnn	Number of gaps
8	2	pp	Number of gaps as a percentage
10	2		Language code (EN or FR)
12	1	'0'	No inhibition of journal
		'1'	Inhibition of journal (no
			journalization of update transactions)
		' '	Retrieval of previous value
13	1	'0'	No inhibition of job follow-up
		'1'	Inhibition of job follow-up
		' '	Retrieval of 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)
21	4	nnnn	Maximum number of accesses for on-line searches in the Database (default value: 300)
25	1	'U'	Implicit update (default option)
		'N'	Explicit update
36	12		PFkeys assigned functions (2).

Position	Length	Value	Meaning
79	1		Dispatch option of Backup:
		'D'	Dispatch: sequential backup of the Database in two separate files.
		'N'	No Dispatch: standard backup of the Database in one PC file.
		' '	Retrieval of previous value

(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 is executed by the system. The PFkey assignment may be viewed on the corresponding sub-menu.

Notes

The Database characteristics remain unchanged if there is no input.

Any field left blank will take on the current options.

The limit of on-line accesses to the Journal depends on the number specified as input to 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 retrieval of archived transactions ('REC' entered on the input parameter line). 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.

Printed report

This procedure prints a report which lists the requested options, associated errors, the number of records restored in the Database for each file 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.

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 have requested the retrieval of the archived transactions.

REST - Description of Steps

User input recognition: PTU010

Code	Physical name	Type	Label
CARTE	User input	Input	User parameters
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database index
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database data
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database users
PAC7PC	Save dir. : PC	Input	Sequential image of the Development Database
PAC7PD	Save dir. : PCI	Input	If Dispatch option of the backup: sequential image 2 of the Database
PAC7PY	Save dir. : PCY	Input	If Dispatch option of the backup: sequential image 3 of the Database
PAC7MB	Tmp dir. : WMB	Output	Parameter

Code	Physical name	Type	Label
PAC7DD	User dir. : RESTDD010	Report	Batch procedures authorization option

Return code:

- 8: no authorization on batch procedure

Validation of journal contents: PTU380

This step is executed only if the Journal file exists.

Code	Physical name	Type	Label
PAC7MB	Tmp dir. : WMB	Input	User transactions
PAC7AE	System - Skel. dir. : AE	Input	Error messages file
PAC7AJ	Journal dir. : AJ	Input	Journal file
PAC7EU	User dir. : EU380	Report	(only if the journal was not archived)

Return codes:

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

Database restoration: PTU400

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

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7MB	Tmp dir. : WMB	Input	User transaction
PAC7PC	Save dir. : PC	Input	Sequential image of the Development Database
PAC7PD	Save dir. : PCI	Input	If backup option Dispatch: sequential image 2 of the Database
PAC7PY	Save dir. : PCY	Input	If Dispatch option in save procedure: sequential image 3 of the Database
PAC7AR	Database dir. : AR	Output	Development Database data
PAC7AN	Database dir. : AN	Output	Development Database index

Code	Physical name	Type	Label
PAC7AY	Database dir. : AY	Output	Development Database extension data
PAC7AJ	Journal dir. : AJ	Output	Development Database Journal
PAC7PS	Tmp dir. : WPS	Output	Work file (2 records, length=144)
PAC7EU	User dir. : RESTEU400	Report	Backup report
PAC7DD	User dir. : RESTDD400	Report	Batch procedures 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.

Warning

This step is REQUIRED to obtain a consistent Database.

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7MB	Tmp dir. : WMB	Input	User transactions
PAC7AR	Database dir. : AR	Input/ Output	Development Database data
PAC7JO	Save dir. : PJ	Input	Journal to apply
PAC7PS	Tmp dir. : WPS	Input	Work file
PAC7OJ	Tmp dir. : WOJ	Output	Update transactions (length=170)
PAC7EU	User dir. : EU420	Report	Retrieval report

Return codes:

- 0: Transactions to be retrieved
- 4: No transactions to be retrieved or error on user input.

In case of abnormal end, the update cannot be performed.

Update of the Administration Database: PACA15

Code	Physical name	Type	Label
PAC7AR	Database dir. : AR	Output	Development Database data

Code	Physical name	Type	Label
PAC7AN	Database dir. : AN	Output	Development Database index
PAC7AY	Database dir. : AY	Output	Development Database extension
PAC7AJ	Journal dir. : AJ	Output	Development Database Journal
PAC7AE	System - Skel. dir.: AE	Input	Error messages
PACGGN	Admin. Database - Base dir.: AN	Input	Administration Database index
PACGGR	Admin. Database - Base dir.: AR	Input	Administration Database data
PACGGY	Admin. Database - Base dir.: AY	Input	Administration Database extension
PACGGU	Admin. Database - Base dir.: GU Base dir.: DC	Input	Administration Database users
PAC7ME	NUL	Input	Work file
PAC7MV	Tmp dir.: WOJ	Input	Update transactions
PAC7RB	NUL	Output	UPDT erroneous transactions (length=80)
PAC7RY	NUL	Output	UPDP erroneous transactions (length=310)
PAC7IE	User dir. : IEA15	Report	Update report (length=132)
PAC7IF	User dir. : IFA15	Report	List of erroneous transactions (length=132)

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

Return codes:

- 0: OK, no error
- 4: Error

REST - Execution Script

```
' -------------------------------------------------------
'      VISUALAGE PACBASE
'
' -------------------------------------------------------
'         - RELOADING RESTORATION OF THE DATABASE -
'
' -------------------------------------------------------
'
' INPUT
```

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

Call Msg_Log (Array("1029" ))
'----------------------------
Call StateList (base, statusL)

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

Dim Ret420
Ret420 = 0

' No Rollback
WshEnv("BVPRB") = "N"

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"
Call BvpEnv("PTU010","PAC7DD",Rep_USR & "\RESTDD010.txt")
Call BvpEnv("PTU010","PAC7MB",Rep_TMP & "\WMB.tmp")
WshEnv("PAC7PC") = BVP_SvName & ""
WshEnv("PAC7PD") = BVP_SvName & "I"
WshEnv("PAC7PY") = BVP_SvName & "Y"
Call RunCmdLog ("BVPTU010")
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"
Call BvpEnv("PTU380","PAC7EU",Rep_USR & "\RESTEU380.txt")
Call BvpEnv("PTU380","PAC7MB",Rep_TMP & "\WMB.tmp")
Call RunCmdLog ("BVPTU380")
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"
Call BvpEnv("PTU400","PAC7EU",Rep_USR & "\RESTEU400.txt")
Call BvpEnv("PTU400","PAC7MB",Rep_TMP & "\WMB.tmp")
WshEnv("PAC7PC") = BVP_SvName & ""
WshEnv("PAC7PD") = BVP_SvName & "I"
Call BvpEnv("PTU400","PAC7PS",Rep_TMP & "\WPS.tmp")
WshEnv("PAC7PY") = BVP_SvName & "Y"
Call RunCmdLog ("BVPTU400")
WshVolEnv("RC") = Return
Call Err_Cod(Return , 0 , "PTU400")

If Not FSO.FileExists( Rep_SAVE & "\PJ"  ) Then
        Call Msg_Log (Array("1004", "PJ in SAVE"))
        Return = 1
        WshVolEnv("RC") = Return
        Wscript.Quit (Return)
End If
```

```
Call Msg_Log (Array("1022" , "PTU420"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AR") =  Rep_BASE & "\AR"
WshEnv("PAC7JO") =  Rep_SAVE & "\PJ"
Call BvpEnv("PTU420","PAC7MB",Rep_TMP & "\WMB.tmp")
Call BvpEnv("PTU420","PAC7OJ",Rep_TMP & "\WOJ.tmp")
Call BvpEnv("PTU420","PAC7PS",Rep_TMP & "\WPS.tmp")
Call BvpEnv("PTU420","PAC7EU",Rep_USR & "\RESTEU420.txt")
Call RunCmdLog ("BVPTU420")
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"
Call BvpEnv("PACA15","PAC7IE",Rep_USR & "\RESTIEA15.txt")
Call BvpEnv("PACA15","PAC7IF",Rep_USR & "\RESTIFA15.txt")
WshEnv("SEMLOCK") = Rep_BASE & "\LO"
WshEnv("SEMADMIN") = Rep_ABASE & "\LO"
Call BvpEnv("PACA15","PAC7MV",Rep_TMP & "\WOJ.tmp")
WshEnv("PAC7ME") = Rep_TMP & "\NUL.tmp"
'PAC7ME not used, on default
WshEnv("PAC7RB") = Rep_TMP & "\NULRB.tmp"
'PAC7RB not used, on default
WshEnv("PAC7RY") = Rep_TMP & "\NULRY.tmp"
'PAC7RY not used, on default
Call RunCmdLog ("BVPACA15")
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


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 managed in on-line mode, from a System backup obtained through a utility and completed by the SASY procedure.

It complements the restoration of the Data (AR), Index (AN), and Extension (AY) files performed by a system utility, by reinitializing the Journal (AJ) file.

Through the RESY procedure, the archived transactions can be retrieved 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 the AN, AR and AY files have been retrieved by the on-site system utility.

On-line access must be closed.

Abnormal execution

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

Printed report

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

Results

Once the procedure has been executed, the network 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 - Input / Processing / Results

A '*' line with user code and password.

The input has the following structure:

Position	Length	Value	Meaning
2	1	'Y'	Line code
10	2		Language code (EN or FR)
12	1	'0'	No journal inhibition
		'1'	Inhibition of journal (update
			transactions are not journalized)
		' '	Retrieval of the last value
14	3	'REC'	If archived transactions are to be retrieved
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 number of accesses for on-line searches in the Database (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.

Position	Length	Value	Meaning
		'N'	No Dispatch Standard backup on a single PC file.
		' '	Same as previous execution.

(1) 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.

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

If there is no input, the Database characteristics don't change.

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	Physical name	Type	Label
CARTE	User input	Input	Parameter
PAC7MB	Tmp dir. : WMB	Output	Parameter

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database index
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database data
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database users
PAC7AR	Database dir. : AR	Input	Development Database data
PAC7DD	User dir. : RESYDD004	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	Physical name	Type	Label
PAC7MB	Tmp dir. : WMB	Input	User transactions
PAC7AE	System - Skel. dir. : AE	Input	Error messages file
PAC7AJ	Journal dir. : AJ	Input	Journal file
PAC7EU	User dir. : EU380	Report	(only if the journal was not archived)

Return codes:

- 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	Physical name	Type	Label
PAC7AR	Database dir. : AR	Output	Data file
PAC7AE	System - Skel dir. : AE	Input	Error messages

Code	Physical name	Type	Label
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database index
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database data
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database users
PAC7MB	Tmp dir. : WMB	Input	User transactions
PAC7PS	Tmp dir. : WPS	Output	Work file (2 records, length=144)
PAC7GZ	User dir. : RESYGZ402	Report	Restoration report

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.

Warning

This step is REQUIRED to obtain a consistent Database.

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7MB	Tmp dir. : WMB	Input	User transactions
PAC7AR	Database dir. : AR	Input/ Output	Development Database data
PAC7JO	Save dir. : PJ	Input	Journal to apply
PAC7PS	Tmp dir. : WPS	Input	Work file
PAC7OJ	Tmp dir. : WOJ	Output	Update transactions (length=170)
PAC7EU	User dir. : EU420	Report	Retrieval report

Return codes:

- 0: Transactions to be retrieved
- 4: No transactions to be retrieved or error on user input.

In case of abnormal end, the update cannot be performed.

Update of the Administration Database: PACA15

Code	Physical name	Type	Label
PAC7AR	Database dir. : AR	Output	Development Database data
PAC7AN	Database dir. : AN	Output	Development Database index
PAC7AY	Database dir. : AY	Output	Development Database extension
PAC7AJ	Journal dir. : AJ	Output	Development Database Journal
PAC7AE	System - Skel. dir.: AE	Input	Error messages
PACGGN	Admin. Database - Base dir.: AN	Input	Administration Database index
PACGGR	Admin. Database - Base dir.: AR	Input	Administration Database data
PACGGY	Admin. Database - Base dir.: AY	Input	Administration Database extension
PACGGU	Admin. Database - Base dir.: GU Base dir.: DC	Input	Administration Database users
PAC7ME	NUL	Input	Work file
PAC7MV	Tmp dir.: WOJ	Input	Update transactions
PAC7RB	NUL	Output	UPDT erroneous transactions (length=80)
PAC7RY	NUL	Output	UPDP erroneous transactions (length=310)
PAC7IE	User dir. : IEA15	Report	Update report (length=132)
PAC7IF	User dir. : IFA15	Report	List of erroneous transactions (length=132)

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

Return codes:

- 0: OK, no error
- 4: Error

RESY - Execution Script

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

```
' ------------------------------------------------------
'
'
' INPUT
' COL  2     : "Y"
' COL 10-11  : INITIAL LANGUAGE CODE (FR, EN)
' COL 12     : "1" INHIBITION OF TRANSACTION LOG
' COL 14-16  :"REC" FOR RECOVERY OF ARCHIVED TRANSACTIONS
' COL 17-20  : 4 CHARACTERS TO BE DISPLAYED ON ALL
'                SCREEN OF THE PRODUCT
' COL 21-24  : "NNNN" MAXIMUM NUMBER OF SEARCH ACCESSES
'              TO THE DATABASE(LISTS)-(DEFAULT VALUE:300)
' COL 25     : "U" (DEFAULT VALUE) : IMPLICIT UPDATE
'            : "N" EXPLICIT UPDATE
' COL 26-29  : CKECKPOINT FREQUENCY
' COL 36-47  : PF-KEYS SIGNIFICATIONS
' COL 79     : BACKUP FILES DISPATCH
'            : "N" (DEFAULT VALUE) : NO DISPATCH (1 FILE)
'            : "D" : DISPATCH (2 FILES)
'
' IF THE JOURNAL FILE OF TRANSACTIONS ON DISK (AJ) IS NOT
' REINITIALIZED, THE RESTORE CHAIN IS NOT EXECUTED.
' IT IS THEREFORE NECESSARY TO EXECUTE THE ARCH
' PROCEDURE FIRST.
'
' ------------------------------------------------------
'
<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"
Call BvpEnv("PTU004","PAC7DD",Rep_USR & "\RESYDD004.txt")
Call BvpEnv("PTU004","PAC7MB",Rep_TMP & "\WMB.tmp")
Call RunCmdLog ("BVPTU004")
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"
Call BvpEnv("PTU380","PAC7EU",Rep_USR & "\RESYEU380.txt")
Call BvpEnv("PTU380","PAC7MB",Rep_TMP & "\WMB.tmp")
Call RunCmdLog ("BVPTU380")
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"
Call BvpEnv("PTU402","PAC7GZ",Rep_USR & "\RESYGZ402.txt")
Call BvpEnv("PTU402","PAC7MB",Rep_TMP & "\WMB.tmp")
Call BvpEnv("PTU402","PAC7PS",Rep_TMP & "\WPS.tmp")
Call RunCmdLog ("BVPTU402")
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"
Call BvpEnv("PTU420","PAC7MB",Rep_TMP & "\WMB.tmp")
Call BvpEnv("PTU420","PAC7OJ",Rep_TMP & "\WOJ.tmp")
Call BvpEnv("PTU420","PAC7PS",Rep_TMP & "\WPS.tmp")
Call BvpEnv("PTU420","PAC7EU",Rep_USR & "\RESYEU420.txt")
Call RunCmdLog ("BVPTU420")
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"
Call BvpEnv("PACA15","PAC7IE",Rep_USR & "\RESYIEA15.txt")
Call BvpEnv("PACA15","PAC7IF",Rep_USR & "\RESYIFA15.txt")
WshEnv("SEMLOCK") = Rep_BASE & "\LO"
WshEnv("SEMADMIN") = Rep_ABASE & "\LO"
Call BvpEnv("PACA15","PAC7MV",Rep_TMP & "\WOJ.tmp")
WshEnv("PAC7ME") = Rep_TMP & "\NUL.tmp"
'PAC7ME not used, on default
WshEnv("PAC7RB") = Rep_TMP & "\NULRB.tmp"
'PAC7RB not used, on default
WshEnv("PAC7RY") = Rep_TMP & "\NULRY.tmp"
'PAC7RY not used, on default
Call RunCmdLog ("BVPACA15")
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 Archiving

ARCH - Introduction

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

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

The archived transactions 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 execution

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 saved.

ARCH - Input / Processing / Results

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

This procedure includes specific optional input to:

- Deactivate the previously archived transactions that are considered obsolete.
- Indicate the absence of previously archived transactions in input.
- Indicate the unavailability of the Data file (AR) in input.
- Request only the re-initialization of the transaction file.

The structure of this input is as follows:

Position	Length	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 in output.

The session number and the date are exclusive. They are ignored if the absence of input transactions is detected (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.

Warning

In this case, the transactions which were already archived are not copied to the output archived transactions file.

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

Recommendations

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 restoration 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 is lost or has been flagged as 'inconsistent', a 'D' in column 16 means that the backup procedure will not take the Data file 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 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 is lost.
- If the archived transactions file is lost or destroyed, an 'I' must be entered in column 15. As a result, the archiving procedure reformats a new sequential backup file of (archived) transactions and the previous file is lost.

You are strongly advised against indicating these options in operation JCLs. It is recommended to duplicate them in a temporary JCL before the execution.

If one of these columns is accidentally set, and if the archiving 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) files to obtain (+1) file.
- 'D' in col. 16: The archiving procedure must be re-executed BEFORE any update on the Database. If an update is performed, the Database will have to be restored completely. However the freeze transactions will be lost.
- '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 which states 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 on-line transactions 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

The first archiving of the Database runs smoothly if the PJ file, which contains archived transactions used as input to the procedure, is created as an empty file, and stored in the SAVE directory during installation.

Archived-transaction purge

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

- 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.
- The user wishes to keep purged archives in the PJ file: the file with PAC7PQ as internal name must be assigned, and it must correspond to a disk file.

  For instance:

  WshEnv("PAC7PQ) = Rep_SAVE & "\PQ"

Archiving 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	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database index
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database data
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database users
PAC7JP	Save dir. : PJ	Input	Previously archived transactions
PAC7AJ	Journal dir. : AJ	Input	Development Database journal to re-initialize
PAC7MB	User input	Input	User transactions

Code	Physical name	Type	Label
PAC7BM	Tmp dir. : WBM	Output	User transactions
PAC7AR	Database dir. : AR	Input/Output	Development Database data
PAC7PJ	Save dir. : PJ-new	Output	Archived update transactions
PAC7PQ	NUL	Output	Deactivated transactions (length=170): : modify the file name to keep the transactions
PAC7EU	User dir. : ARCHEU300	Report	Output report
PAC7DD	User dir. : ARCHDD300	Report	Batch procedures authorization option

Return codes:

- 0: No error detected on the files,
- 4: Error in journal record (Date or session number not numeric)
- 8: No access authorization for batch procedure, OR: invalid Database; in this case, restart the procedure with 'D' in column 16 of the user input .
- 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	Physical name	Type	Label
PAC7BM	Tmp dir. : WBM	Input	User transactions
PAC7AR	Database dir. : AR	Input/ Output	Data file
PAC7AE	System - Skel dir. : AE	Input	Error messages file
PAC7AJ	Journal dir. : AJ	Output	Journal file to re-initialize
PAC7EU	User dir. : ARCHEU320	Report	Re-initialization report

Return codes:

- 0: No error detected,
- 8: The Database is invalid.

If the archiving and backup procedures are grouped into one job, the PTU320 return code can be tested in order to condition the execution of the backup programs.

ARCH - Execution Script

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

<script language="VBScript">
Dim MyProc
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("1029" ))
'----------------------------
Call StateList (base, statusL)

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") = Rep_TMP & "\NULPQ.tmp"
'PAC7PQ not used
Call BvpEnv("PTU300","PAC7BM",Rep_TMP & "\WBM.tmp")
Call BvpEnv("PTU300","PAC7EU",Rep_USR & "\ARCHEU300.txt")
Call BvpEnv("PTU300","PAC7DD",Rep_USR & "\ARCHDD300.txt")
Call RunCmdLog ("BVPTU300")

 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 , 4 , "PTU300")



Call Msg_Log (Array("1022" , "PTU320"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PAC7AJ") = Rep_JOURNAL & "\AJ"
Call BvpEnv("PTU320","PAC7BM",Rep_TMP & "\WBM.tmp")
Call BvpEnv("PTU320","PAC7EU",Rep_USR & "\ARCHEU320.txt")
Call RunCmdLog ("BVPTU320")

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 recognizing each deletion, and sorting the data again according to the most frequent access order.

It uses one or three (if Dispatch option) backup files of the Development Database to rebuild one or three (if Dispatch option) sequential image(s). This resulting image file must then be used as input to 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.

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.

The deletions recognized by the reorganization may have been made logically upon the Database update, or generated by utilities. For example:

- Deletion of unused Production sessions (SCM Module),
- Deletion of unused entities, determined by the unused-entity extraction utility, EXPU (see the PACX procedure in 'the Developer's Procedures' manual).

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.

Abnormal execution

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

As specified in the '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 - Input / Processing / Results

A '*' line with user code and password.

Specific user input for the procedure (optional) which enables you to specify:

- Libraries to be purged,
- sessions to be purged or to be kept,
- users to be purged,
- entities to be purged (notion of cross-references taken into account),
- a report of the duplicate indexes of the REOR procedure.

Position	Length	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.

Warning: Contrary to the MLIB procedure, the REOR procedure does not perform any check on dependent libraries in case of a purge; you must then be very careful and purge, if necessary, the dependent libraries before the parent library to prevent any inconsistency in the Database.

Position	Length	Value	Meaning
2	1	'V'	Purge frozen sessions
		'S'	Save frozen sessions
			'V' and 'S' lines are not compatible

Position	Length	Value	Meaning
3		ssss	Session number(s): up to 17 session numbers per line, entered without any separator.

Maximum number of sessions indicated on the request: 999.

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

Position	Length	Value	Meaning
2	1	'U'	Purge users (up to 9 user codes per line)

The purge of a user deletes this user and all its uses (GP lines and locks) in the Development Database. It is independent of the deletion of a user code in the Administration Database.

Maximum number of users to be purged: 100.

Position	Length	Value	Meaning
2	1	'E'	Physical purge of entities (transactions provided by EXPU)
3			Entity (required)
	1		.Type
	2		.UE call type (if Type "$")
6	30		Code of the entity to be purged (this code can be generic)
36	3		Library code (required)
39			Flag for purge of lower-level Libraries
		' ' or 'Y'	Purge of the Library and of its sub-Libraries
		'N'	Purge of the Library alone

One line per entity. This command allows to purge the User Entity and the associated bulk data. The 'List of 'purged' entities' indicates what has been done.

In case of a generic request, the entity code must be completed with * to make up for six characters. If the code contains six '*', all of the User Entities are deleted.

Position	Length	Value	Meaning
2	1	'P'	Physical purge of 'bulk' entities
3			Entity (required)
	1		Type
	2		UE call type (if type '$')
6	30		Code of the entity to be purged (required, complete identifier)
36	2		Entity Description type (required, value '00' not authorized
		nn	Purge of the 'nn' descriptions of the entity
		'**'	Purge of all Descriptions of the entity
38	3		Library code (required)
		'***'	All Libraries of the entity
41	1		Flag for purge of lower-level Libraries
		' ' or 'Y'	Purge of the Library and its sub-Libraries
		'N'	Purge of the Library alone
42	4		Session lower bound (one of the two bounds is required)
		' '	Purge from the lower bound
		nnnn	Purge from session 'nnnn' (included)
		'****'	Purge of all sessions
46	4		Session higher bound (one of the two bounds is required)
		' '	Purge up to the higher bound
		mmmm	Purge up to session 'mmmm' (included)
		'****'	Purge in all sessions

One line per 'bulk'-type entity. This command enables to purge the 'bulk' data of the attached files created via Developer workbench. This purge is performed between a start session and an end session.

Note on 'E- and P-type' purge requests: The execution of the REOR procedure processes a maximum number of 2,500 instances of an entity type other than a User Entity and a number of 1,000 instances of the User Entity type.

Position	Length	Value	Meaning
2	1	'D'	List of duplicate indexes of the REOR procedure
3	1	' '	no report of duplicate indexes
		'1'	Report of duplicate indexes
		'2'	Report of PMS calls which might be incoherent
		'3'	Reports of values '1' and '2'

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 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 report

This procedure prints a report which lists the 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 restoration 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.
- The spaces allocated to the sorts 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	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database index
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database data
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database users
PAC7PC	Save dir. : PC	Input	Sequential image of the Development Database
PAC7MB	User Input	Input	Input work file
PAC7BM	Tmp dir. : WBM	Output	Formatted records
PAC7PU	Tmp dir. : WPU	Output	Entity records to be purged (length=44)
PAC7EE	User dir. : REOREE2CL	Report	Control report
PAC7DD	User dir. : REORDD2CL	Report	Batch procedures authorization option

.Return code(s):

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

Retrieval of data: PTU200

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

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7BM	Tmp dir. : WBM	Input	User transactions
PAC7PC	Save dir. : PC	Input	Sequential image of the Development Database
PAC7PY	Save dir. : PCY	Input	Sequential image of the extension data of the Development Database
PAC7PR	Tmp dir. : WPR	Output	Formatted records (length=176 size= ND)
PAC7NX	Tmp dir. : WNX	Output	Long data
PAC7NY	Tmp dir. : WNY	Output	Bulk data
PAC7AU	Tmp dir. : WAU	Output	PR image (length=153)
PAC7EE	User dir. : REOREE200	Report	Report on statistics retrieval

ASCII Sort: PTU205

Code	Physical name	Type	Label
PAC7PR	Tmp dir. : WPR	Input	Sorted data records
PAC7RP	Tmp dir. : WRP	Output	ASCII sorted data records

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

Purge: PTU210

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

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7PR	Tmp dir. : WRP	Input	Sorted records
PAC7BM	Tmp dir. : WBM	Input	User transactions
PAC7PU	Tmp dir. : WPU	Input	Entity records to be purged
PAC7QS	Tmp dir. : WQS	Output	Purged records (length=176, size= ND)
PAC7UM	Tmp dir. : WUM	Output	Macro-structure call lines (length=176)

Code	Physical name	Type	Label
PAC7EE	User dir. : REOREE210	Report	Library and session purge report
PAC7EK	User dir. : REOREK210	Report	Entity-purge report
PAC7EB	User dir. : REOREB210	Report	Technical report

Return codes:

- 0: OK
- 8: Capacity overflow

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

Index rebuilding: PTU220

This step executes two types of processing:

- Reconstruction of the indexes using the data
- Separation of the current session and the frozen sessions.

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages file
PAC7BM	Tmp dir. : WBM	Input	User transactions
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database index
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database data
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database users
PAC7UR	Tmp dir. : WQS	Input	Purged data
PAC7NX	Tmp dir. : WNX	Input	Long data
PAC7UM	Tmp dir. : WUM	Input	Macro-structure call lines
PAC7PA	Tmp dir. : WPA	Output	Data from frozen sessions (length=153 size=NH)
PAC7PB	Tmp dir. : WPB	Output	Data from current session (length=153 size=NC)
PAC7PC	Tmp dir. : WPC	Output	First data records (length=153)
PAC7AN	Tmp dir. : WAN	Output	Intermediate index file (length=60 size=NI)

Code	Physical name	Type	Label
PAC7MR	Tmp dir. : WMR	Input/ Output	Macro-structure call lines
PAC7EE	User dir. : REOREE220	Report	Index-building report

ASCII Sort: PTU225

Code	Physical name	Type	Label
PAC7AN	Tmp dir.: WAN	Input	Sorted index
PAC7NA	Tmp dir.: WNA	Output	ASCII Sorted index

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

Processing of extension data: PTU226

Code	Physical name	Type	Label
PAC7NY	Tmp dir. : WNY	Input	Bulk data
PAC7PA	Tmp dir. : WPA	Input	Data from frozen sessions
PAC7PB	Tmp dir. : WPB	Input	Data from current session
PAC7PC	Tmp dir. : WPC	Input	First data record
PAC7QA	Tmp dir. : WQA	Output	Data from frozen sessions (length=153)
PAC7QB	Tmp dir. : WQB	Output	Data from current session (length=153)
PAC7QC	Tmp dir. : WQC	Output	First data record length=153)
PAC7QY	Tmp dir. : WQY	Output	Long data (length=1,018)

Merge: PTU240

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

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7AN	Tmp dir. : WNA	Input	Sorted index
PAC7AU	Tmp dir. : WAU	Input	PR image
PAC7BM	Tmp dir. : WBM	Input	User transactions

Code	Physical name	Type	Label
PAC7PA	Tmp dir. : WQA	Input	Data from frozen sessions
PAC7PB	Tmp dir. : WQB	Input	Data from current session
PAC7PC	Tmp dir. : WQC	Input	First data records
PAC7QY	Tmp dir. : WQY	Input	Extension data
PAC7CP	Save dir. : PC-new	Output	Sequential image of the Development Database
PAC7PD	Save dir. : PCI-new	Output	Sequential image of the Development Database
PAC7PY	Save dir. : PCY-new	Output	Sequential image of the Development Database
PAC7IE	User dir. : REORIE240	Report	Building of Logical Database

Deletion of long data file: IDCAMS

This step executes a DELETE of the work file which contains the long data.

Deletion of bulk data file: IDCAMS

This step executes a DELETE of the work file which contains the bulk data.

REOR - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'                - REORGANIZATION OF THE DATABASE -
'
' ------------------------------------------------------
'
' THE REOR PROCEDURE MAY BE USED IN TWO CASES:
' . WHEN PART OF THE DATA WAS DELETED BECAUSE OF A MAL-
'  FUNCTION 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;
'
' ------------------------------------------------------
'
<job id=REOR>

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

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

<script language="VBScript">

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

Call ExistsSV(BVP_SvName & "")
Call ExistsSV(BVP_SvName & "I")
Call ExistsSV(BVP_SvName & "Y")

Call Msg_Log (Array("1022" , "PTU2CL"))
'-----------------------------------
'Example of Input File extracted from PACX/EXPU :
' Call BvpEnv("PTU2CL","PAC7MB",RepT_USR & "\PACXMR.txt")
' The first line must contain User/Password information
'With  RepT_USR is Global User Directory.

WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7MB") = Fic_Input
Call BvpEnv("PTU2CL","PAC7BM",Rep_TMP & "\WBM.tmp")
Call BvpEnv("PTU2CL","PAC7PU",Rep_TMP & "\WPU.tmp")
Call BvpEnv("PTU2CL","PAC7DD",Rep_USR & "\REORDD2CL.txt")
Call BvpEnv("PTU2CL","PAC7EE",Rep_USR & "\REOREE2CL.txt")
WshEnv("PAC7PC") = BVP_SvName & ""
Call RunCmdLog ("BVPTU2CL")
If Return = 4 Then
Call Msg_Log (Array("1057"))
End If
If Return = 8 Then
Call Msg_Log (Array("1028"))
End If
Call Err_Cod(Return , 0 , "PTU2CL")

Call Msg_Log (Array("1022" , "PTU200"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7PC") = BVP_SvName & ""
WshEnv("PAC7PY") = BVP_SvName & "Y"
Call BvpEnv("PTU200","PAC7BM",Rep_TMP & "\WBM.tmp")
Call BvpEnv("PTU200","PAC7PR",Rep_TMP & "\WPR.tmp")
Call BvpEnv("PTU200","PAC7NX",Rep_TMP & "\WNX.tmp")
Call BvpEnv("PTU200","PAC7NY",Rep_TMP & "\WNY.tmp")
Call BvpEnv("PTU200","PAC7AU",Rep_TMP & "\WAU.tmp")
Call BvpEnv("PTU200","PAC7EE",Rep_USR & "\REOREE200.txt")
Call RunCmdLog ("BVPTU200")
Call Err_Cod(Return , 0 , "PTU200")

Call Msg_Log (Array("1022" , "PTU205"))
'-----------------------------------
Call BvpEnv("PTU205","PAC7PR",Rep_TMP & "\WPR.tmp")
Call BvpEnv("PTU205","PAC7RP",Rep_TMP & "\WRP.tmp")
Call RunCmdLog ("BVPTU205")
```

```
Call Err_Cod(Return , 0 , "PTU205")
'OK :
Call DelFile (Rep_TMP & "\WPR.tmp")

Call Msg_Log (Array("1022" , "PTU210"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
Call BvpEnv("PTU210","PAC7EB",Rep_USR & "\REOREB210.txt")
Call BvpEnv("PTU210","PAC7EE",Rep_USR & "\REOREE210.txt")
Call BvpEnv("PTU210","PAC7EK",Rep_USR & "\REOREK210.txt")
Call BvpEnv("PTU210","PAC7BM",Rep_TMP & "\WBM.tmp")
Call BvpEnv("PTU210","PAC7PR",Rep_TMP & "\WRP.tmp")
Call BvpEnv("PTU210","PAC7PU",Rep_TMP & "\WPU.tmp")
Call BvpEnv("PTU210","PAC7QS",Rep_TMP & "\WQS.tmp")
Call BvpEnv("PTU210","PAC7UM",Rep_TMP & "\WUM.tmp")
Call RunCmdLog ("BVPTU210")
If Return = 8 Then
Call Msg_Log (Array("1056"))
End If
If Return = 12 Then
Call Msg_Log (Array("1026"))
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"
Call BvpEnv("PTU220","PAC7BM",Rep_TMP & "\WBM.tmp")
Call BvpEnv("PTU220","PAC7AN",Rep_TMP & "\WAN.tmp")
Call BvpEnv("PTU220","PAC7MR",Rep_TMP & "\WMR.tmp")
Call BvpEnv("PTU220","PAC7NX",Rep_TMP & "\WNX.tmp")
Call BvpEnv("PTU220","PAC7PA",Rep_TMP & "\WPA.tmp")
Call BvpEnv("PTU220","PAC7PB",Rep_TMP & "\WPB.tmp")
Call BvpEnv("PTU220","PAC7PC",Rep_TMP & "\WPC.tmp")
Call BvpEnv("PTU220","PAC7UM",Rep_TMP & "\WUM.tmp")
Call BvpEnv("PTU220","PAC7UR",Rep_TMP & "\WQS.tmp")
Call BvpEnv("PTU220","PAC7EE",Rep_USR & "\REOREE220.txt")
Call RunCmdLog ("BVPTU220")
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"))
'-----------------------------------
Call BvpEnv("PTU225","PAC7AN",Rep_TMP & "\WAN.tmp")
Call BvpEnv("PTU225","PAC7NA",Rep_TMP & "\WNA.tmp")
```

```
Call RunCmdLog ("BVPTU225")
Call Err_Cod(Return , 0 , "PTU225")

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

Call Msg_Log (Array("1022" , "PTU226"))
'-----------------------------------
Call BvpEnv("PTU226","PAC7PA",Rep_TMP & "\WPA.tmp")
Call BvpEnv("PTU226","PAC7PB",Rep_TMP & "\WPB.tmp")
Call BvpEnv("PTU226","PAC7PC",Rep_TMP & "\WPC.tmp")
Call BvpEnv("PTU226","PAC7QA",Rep_TMP & "\WQA.tmp")
Call BvpEnv("PTU226","PAC7QB",Rep_TMP & "\WQB.tmp")
Call BvpEnv("PTU226","PAC7QC",Rep_TMP & "\WQC.tmp")
Call BvpEnv("PTU226","PAC7QY",Rep_TMP & "\WQY.tmp")
Call BvpEnv("PTU226","PAC7NY",Rep_TMP & "\WNY.tmp")
Call RunCmdLog ("BVPTU226")
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"
Call BvpEnv("PTU240","PAC7AN",Rep_TMP & "\WNA.tmp")
Call BvpEnv("PTU240","PAC7AU",Rep_TMP & "\WAU.tmp")
Call BvpEnv("PTU240","PAC7BM",Rep_TMP & "\WBM.tmp")
Call BvpEnv("PTU240","PAC7PA",Rep_TMP & "\WQA.tmp")
Call BvpEnv("PTU240","PAC7PB",Rep_TMP & "\WQB.tmp")
Call BvpEnv("PTU240","PAC7PC",Rep_TMP & "\WQC.tmp")
Call BvpEnv("PTU240","PAC7QY",Rep_TMP & "\WQY.tmp")
WshEnv("PAC7CP") = BVP_SvName & "-new"
WshEnv("PAC7PD") = BVP_SvName & "I-new"
WshEnv("PAC7PY") = BVP_SvName & "Y-new"
Call BvpEnv("PTU240","PAC7IE",Rep_USR & "\REORIE240.txt")
Call RunCmdLog ("BVPTU240")
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(BVP_SvName & "")
Call Turnover(BVP_SvName & "I")
Call Turnover(BVP_SvName & "Y")
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 Development Database via a PAF extractor (selection of criteria).

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

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

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

Execution conditions

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

PACX - Input Common to Extractors

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

Position	Length	Value	Meaning
		'2'	CPSN : formatting for UPDP with explicit 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 with user code ('*'-line user code)
		'1'	No lock extraction
		'2'	Lock extraction wth user code (original user code)
		'N'	For RMEN only : no extraction of locked entities by an other user
50	1	' '	No transfer of password
		'1'	Password transfer
69	3	bbb	Library code for the '*'-line of the output file(s) (For EXTR, EXLI and EXUE only)
76	5	nnnnT	Session number for the '*'-line of the output file(s) (For EXTR, EXLI and EXUE only)

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

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

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

Important

- One extractor type only for each run: If the procedure detects more than one type of extractors, it will take only the first one into account.
- The formatting type of the first '*' line only is taken into account.
- Formatting for CPSN: This procedure is part of the 'Partitioned Database Manager' optional utility. Its use is therefore subject to a separate purchase agreement.
- Maximum number of input '*' lines : 1 for RMEN and EXPJ, 1000 for EXTR, EXTA, EXUE and EXPU.

Results

The PACX procedure produces:

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

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

Password transfer option ('*'-line col. 50 = 1): for Administrators only.

EXPJ - User Input:

User input specific to this procedure. It specifies the extraction characteristics.

Position	Length	Value	Meaning
2	1	'J'	Line code
3	1	'S'	Selection on session number
		'D'	Selection on date
4	1	' '	Chronological sort
		'N'	No chronological sort
5	1	' '	Sort by user
		'N'	No sort by user
6	1	' '	Sort by Library
		'N'	No sort by library
7	1	' '	Sort by session
		'N'	No sort by session
8	8	uuuuuuuu	User code (carried over to the assign line of the result file)
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		Version of selected transactions
		' '	Selection of all sessions
		'Z'	Selection of current session
		'T'	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 a continuation line

The user code specified in this input line is not included in the selection.

Second user input if selection on user code:

Position	Length	Value	Meaning
2	1	'J'	Line code
3	1	'*'	Indicates a continuation line
4	8	uuuuuuuu	User code to be selected

This continuation line is unique ; the selection is performed on one user only.

This procedure recognizes one input only, i.e. the first input line followed or not by a continuation line.

An error message is sent when there are multiple input lines.

Reports

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

Results

If you requested the conversion of the Journal entries into transactions, the result of the EXPJ procedure is a sequential file which contains all selected transactions.

Extraction of Libraries

EXLI - Introduction:

The EXLI procedure extracts a complete library or a sub-network of Libraries in a transactions file format.

This will be used, according to the formatting requested, as input to the UPDT or UPDP update procedures or to the CPSN Sub-Network Comparison procedure.

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 - User Input:

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

Printed report

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,
- 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 unused entities in a particular context;
- 'Physical' purge of entities which have never been used.

Terminology

- Final entities

  These are entities which are not used by other entities.
- Free-type cross-reference

  Reference whose existence does not prevent the deletion of the Definition of the Entity on which it depends.

Principles

- Logical purge: the EXPU procedure shows the list of the 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 the 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 - User Input:

One line with the extraction characteristics:

Position	Length	Value	Meaning
2	1	'P'	Line code
3	1		Purge of frozen sessions
		'S'	yes
		' '	No purge
		'P'	Purge of production sessions without entities in production
		'G'	Purge of production sessions which do not contain any entities in production, or else limitation of the number of production-turnover sessions for each entity
4	1		Purge of libraries
		'B'	yes
		' '	No purge

Position	Length	Value	Meaning
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	'1'	Print option for the last entity update: user code and date
		' '	No print option
9	1	'1'	Print option for the last session in which the entity is used
		' '	No print option
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 Programs are processed (information required if the entity type is 'P' or blank)
14	2	cc	Meta Entity call type if User Entities are processed (information required if entity type is '$' or 'Y')
26	1		Number of sessions to take into account for a 'G'-type request (in this case, the most recent n sessions will be taken into account)

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 LIB 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 LIB 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 LIB 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 reports

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 the libraries to be physically purged.
- the list of users whom GP lines are to be physically purged.

Result

The result of this procedure is:

- In the case of a logical purge, a sequential file which contains entity deletion transactions to be used as input to 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 entity type,
– the entity code,
– the library code (see section 'REOR - Input / Processing / Results' in subchapter 'REOR - Reorganization').

Standardization Utility

RMEN - 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 the Dictionary and WorkStation 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.

RMEN - User Input:

Several command lines per entity to be processed:

First line - concerned entity :

Position	Length	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

Position	Length	Value	Meaning
6	1	' '	Line type
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 :

Position	Length	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		For extraction of WorkStation entities: methodology code
		'//A'	SSADM
		'//M'	MERISE
		'//D'	YSM
		'//O'	OMT
		'//F'	IFW
13	3	'ALL'	for 'MV' and 'MR': Selects all the UE of a meta entity or all the segments of a data structure (implicit option for 'RN' and 'RP)
16	6	rrrrrr	Parent Data Element code

Third line - new codes:

Position	Length	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

Position	Length	Value	Meaning
51	8	eeeeeeee	For screens, new map code

Fourth line - selection for REPLACE:

Position	Length	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		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):

Position	Length	Value	Meaning
2	2	'W2'	Line code

Position	Length	Value	Meaning
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):

Position	Length	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 which calls it.
- When a macro-structure is called by a batch Program or on-line Screen, it must be moved o 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 conditions

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 not rejected.

Printed report

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).

Note

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.

Warning: 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 used entities must exist in a library whose level is greater than or equal to that of the target library.

When an occurrence is renamed, if it is called on Assigned Text 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.

Warning: An upward move of a Data Structure involves the upward move of all of its Segments contained in the source library in cases where the 'global upward move' field contains 'ALL'. If this field is blank, the Segments 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 single 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 only affects 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 Data Element field.

A child Data Element can be turned into a parent Data Element or may be assigned another parent by specifying a new parent Data Element code in the parent Data Element field.

In this case, the parent Data Element must be defined in a Library whose level is greater than or equal to the target Library.

A parent Data Element contained in a request must not have been previously processed as a source Data 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' of a program) 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 the:

- called Macro-Structures,
- called 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.

Warning: When the global upward move field contains 'ALL', an upward move of a Meta Entity involves the upward move of all of its User Entities contained in the source Library. If this field is blank, the User Entities remain in the source Library.

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

User Entities

The existence of the Meta entity in a Library higher than or equal to that of the target Library is checked, as is that of occurrences linked to the User Entities or details lines.

Merise entities

For MERISE Objects and Properties called in description lines, 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 '//A', '//M', '//D', '//O' and '//F' 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 code of the source Library 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').

Warning: One procedure execution can process entities related to only one Methodology.

Sub-Network and Entities 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 'slave' sub-network or entities with the 'master' 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 reference entity

Execution conditions

None.

Abnormal execution

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

CPSN - User Input:

No specific line.

Note

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	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7AN	Database dir. : AN	Input	Development Database Index file
PAC7AR	Database dir. : AR	Input	Development Database Data file
PAC7AY	Database dir. : AY	Input	Development Database Extension Data
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database Index file
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database Data file
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database Users
PAC7PJ	Save dir.: PJ	Input	Archived transactions
PAC7MB	User input	Input	User input
PAC7MA	NUL	Input	CPSN Master file
PAC7ES	NUL	Input	CPSN Slave file
PAC7BM	Tmp dir.: WBM	Input/Output	User input
PAC7MM	Tmp dir.: WMM	Input/Output	EXPU Work file
PAC7MJ	Tmp dir.: WMJ	Input/Output	EXPJ Work file
PAC7TE	Tmp dir.: WTE	Input/Output	RMEN Work file
PAC7RE	Tmp dir.: WRE	Input/Output	RMEN Work file
PAC7RM	Tmp dir.: WRM	Input/Output	RMEN Work file
PAC7WD	Tmp dir.: WWD	Input/Output	Extracted transactions
SYSEXT	Tmp dir.: WSY	Input/Output	Indexed Work File
PAC7MV	User dir.: PACXMV	Output	Extracted transactions for UPDT

Code	Physical name	Type	Label
PAC7MR	User dir.: PACXMR	Output	Extracted transactions for REOR (EXPU)
PAC7MX	User dir.: PACXMX	Output	Non extracted entities (PACX)
PAC7GY	User dir.: PACXGY	Output	Extracted transactions for UPDP
PAC7TD	User dir.: PACXTD	Output	Extracted transactions for CPSN
PAC7UE	User dir.: PACXUE	Output	Extracted transactions for EXUE
PAC7IA	User dir.: PACXIA	Report	General printout of the program stream
PAC7DD	User dir.: PACXDD	Report	Errors on input transactions
PAC7ED	User dir.: PACXED	Report	Extractions report
PAC7EE	User dir.: PACXEE	Report	Extractions report
PAC7EG	User dir.: PACXEG	Report	Extractions report
PAC7EM	User dir.: PACXEM	Report	Extractions report
PAC7EP	User dir.: PACXEP	Report	Extractions report
PAC7EQ	User dir.: PACXEQ	Report	Extractions report
PAC7EU	User dir.: PACXEU	Report	Extractions report
PAC7EZ	User dir.: PACXEZ	Report	Extractions report

Return codes:

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

PACX - Execution Script

```
' -------------------------------------------------------
'      VISUALAGE PACBASE
'
' -------------------------------------------------------
'           - EXTRACTIONS FROM DATABASE -
'           - EXTRACTIONS COMPARATOR    -
' -------------------------------------------------------
'
' THE PACX PROCEDURE ALLOWS TO PERFORM VARIOUS TYPES
' OF DATA EXTRACTIONS FROM THE DEVELOPMENT DATABASE
```

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

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

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

<script language="VBScript">

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

</script>
</job>
```

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 or via the Administrator workbench. 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).

ESES - Session Numbers Extraction

ESES - Introduction

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

Preliminary operations

Backup of the Development Database :

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

If Pactables is installed:

- Table backup (SVTA)

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

- DSMS journal archiving (DARC)
- Backup of the DSMS Database (DSAV)

Execution conditions

None.

ESES - User Input

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

One line (optional) per session number to force:

Position	Length	Value	Meaning
2	1	'S'	Line Code
3	4	nnnn	Original session number
7	4	nnnn	New session number

ESES - Description of Steps

Creation of the Session matching file: PTUESS

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7AR	Database dir. : AR	Input	Development Database data
PAC7AN	Database dir. : AN	Input	Development Database index
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database data
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database index
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database users
PAC7MB	User input	Input	Input transactions
PAC7MV	User dir. : MVESES	Output	Session-number correspondence table
PAC7EU	User dir. : ESESEUESS	Report	Extraction report
PAC7DD	User dir. : ESESDDESS	Report	Batch procedure authorization option

Return codes:

- 8: No authorization for this procedure.

ESES - Execution Script

```
' ------------------------------------------------------

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

<script language="VBScript">
Dim MyProc
MyProc = "ESES"
</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" , "PTUESS"))
'--------------------------------
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("PAC7MB") = Fic_Input
WshEnv("PAC7MV") = BVP_EsesFi
Call BvpEnv("PTUESS","PAC7DD",Rep_USR & "\ESESDDESS.txt")
Call BvpEnv("PTUESS","PAC7EU",Rep_USR & "\ESESEUESS.txt")
Call RunCmdLog ("BVPTUESS")

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 - User Input

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:

Position	Length	Value	Meaning
2	1	'S'	Line code
3	21		Code of files to be retrieved (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

Compression of Session numbers: PTUCSS

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

Code	Physical name	Type	Label
PACGGR	Admin Database - DBase dir. : AR	Input	Administration Database data
PACGGN	Admin Database - DBase dir. : AN	Input	Administration Database index
PACGGU	Admin Database - DBase dir. : GU	Input	Administration Database users
PAC7MV	User dir. : MVESES	Input	Session numbers correspondence table
PAC7MB	User input	Input	Parameter line
PAC7PC	Save dir. : PC	Input	Development Database backup
PAC7PD	Save dir. : PCI	Input	If Dispatch option of the backup: Backup 2 of the Development Database
PAC7PY	Save dir. : PCY	Input	If Dispatch option of the backup: Backup 3 of the Development Database
PAC7CP	Save dir. : PC-new	Output	If Dispatch option of the backup:
PAC7DP	Save dir. : PCI-new	Output	If Dispatch option of the backup: Backup 2 of the Development Database
PAC7YP	Save dir. : PCY-new	Output	If Dispatch option of the backup: Backup 3 of the Development Database
PAC7PJ	Save dir. : PJ	Input	Backup of the Development Database Journal
PAC7JP	Save dir. : PJ-new	Output	Backup of the Development Database Journal
PACDBB	DSMS Database dir. : BB	Input	DSMS Database backup (if DSMS is installed)
PACDJB	DSMS Database dir.: BB -new	Output	DSMS Database backup (if DSMS is installed)
PACDDJ	DSMS Database dir. : BJ	Input	Retrieval of DSMS archived Journal (if DSMS is installed)
PACDJD	DSMS Database dir.: BJ-new	Output	Retrieval of DSMS archived Journal (if DSMS is installed)
PAC7TC	Save dir. : TC	Input	Retrieval of Pactables backup (if Pactables is installed)
PAC7CT	Save dir. : TC-new	Output	Retrieval of Pactables backup (if Pactables is installed)

Code	Physical name	Type	Label
PAC7EU	User dir. : CSESEUCSS	Report	Execution output
PAC7DD	User dir. : CSESDDCSS	Report	Batch procedure authorization option

Return codes:

- 8: No authorization for this procedure.

CSES - Execution Script

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

<script language="VBScript">
Dim MyProc
MyProc = "CSES"
</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" , "PTUCSS"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
```

```
WshEnv("PAC7MB") = Fic_Input
WshEnv("PAC7MV") = BVP_EsesFi
If Not FSO.FileExists(BVP_EsesFi) Then
Call Msg_Log (Array("1001" , BVP_EsesFi))
c_error = 1
WshVolEnv("RC") = 32
WshEnv("PAC7MV") = Rep_TMP & "\NUL.tmp"
End If
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7PC") =  BVP_SvName & ""
WshEnv("PAC7CP") =  BVP_SvName & "-new"
WshEnv("PAC7PD") =  BVP_SvName & "I"
WshEnv("PAC7DP") =  BVP_SvName & "I-new"
WshEnv("PAC7PY") = BVP_SvName & "Y"
WshEnv("PAC7YP") =  BVP_SvName & "Y-new"
WshEnv("PAC7PJ") = Rep_SAVE & "\PJ"
WshEnv("PAC7JP") = Rep_SAVE & "\PJ-new"

Call BvpEnv("PTUCSS","PACDBB",Rep_TMP & "\NUL.tmp")
'PACDBB not used, on default
If WshEnv("PACDBB") = Rep_TMP & "\NUL.tmp" Then
       WshEnv("PACDJB") = Rep_TMP & "\NULJD.tmp"
Else
      WshEnv("PACDJB") = WshEnv("PACDBB") & "-new"
End if


Call BvpEnv("PTUCSS","PACDDJ",Rep_TMP & "\NUL.tmp")
'PACDDJ not used, on default
If WshEnv("PACDDJ") = Rep_TMP & "\NUL.tmp" Then
       WshEnv("PACDJD") = Rep_TMP & "\NULJD.tmp"
Else
      WshEnv("PACDJD") = WshEnv("PACDDJ") & "-new"
End if

Call BvpEnv("PTUCSS","PAC7TC",Rep_TMP & "\NUL.tmp")
'PAC7TC not used, on default
If WshEnv("PAC7TC") = Rep_TMP & "\NUL.tmp" Then
       WshEnv("PAC7CT") = Rep_TMP & "\NULCT.tmp"
Else
       WshEnv("PAC7CT") = WshEnv("PAC7TC") & "-new"
End if

Call BvpEnv("PTUCSS","PAC7DD",Rep_USR & "\CSESDDCSS.txt")
Call BvpEnv("PTUCSS","PAC7EU",Rep_USR & "\CSESEUCSS.txt")
Call RunCmdLog ("BVPTUCSS")

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(BVP_SvName & "")
Call Turnover(BVP_SvName & "I")
Call Turnover(BVP_SvName & "Y")
Call Turnover(Rep_SAVE & "\PJ")
If  FSO.FileExists(WshEnv("PACDBB")) _
    and WshEnv("PACDBB") <> Rep_TMP & "\NUL.tmp" Then
    Call Turnover(WshEnv("PACDBB"))
end if

If  FSO.FileExists(WshEnv("PACDDJ")) _
    and WshEnv("PACDDJ") <> Rep_TMP & "\NUL.tmp" Then
    Call Turnover(WshEnv("PACDDJ"))
end if
If  FSO.FileExists(WshEnv("PAC7TC")) _
    and WshEnv("PAC7TC") <> Rep_TMP & "\NUL.tmp" Then
    Call Turnover(WshEnv("PAC7TC"))
end if

End If

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

</script>
</job>
```

Management of Access Keys and User Rights

UKD0 - Introduction

This procedure prepares the matching file for the user rights of the old and the new key, before the key is activated via the UKD1 procedure.

The output file must be checked by the user : it may be modified, especially if the key range has changed.

Execution condition

Non, since this procedure does not update the database.

Printed report

This procedure prints a report which lists the errors (EK2) as well as a validation report (EK3) which allows, if necessary, to correct the generated matching file.

Result

A matching file for the user rights between the old and the new key. This file will be used as input to the UKD1 procedure.

UKD0 - User input

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

The user must be an Administrator.

A 'K'-type line to specify the key to be activated.

This line is optional ; if it it not specified, the first key whose expiring date is still valid will be taken into account.

Pos.	Len.	Value	Meaning
2	1	'K'	Line code to activate the key
3	14		Identifier of the key to be activated (formatted as CCYYMMDDHHMMSS). This key must exist in the Administration Database as 'Archive', must not be a test key and must not be expired.

UKD0 - Description of Steps

Preparation of the Rights' matching file: PTUKD0

Code	Physical name	Type	Label
PAC7MB		Input	User input
PAC7AR	Admin Database - DBase dir.: AR	Input	Administration Database data
PAC7AN	Admin Database - DBase dir.: AN	Input	Administration Database index
PAC7AE	System - Skel. dir. : AE	Input	Error messages
PACGGR	Admin Database - DBase dir.: AR	Input	Administration Database data
PACGGN	Admin Database - DBase dir.: AN	Input	Administration Database index
PACGGU	Admin Database - DBase dir. : GU	Input	Administration Database users
PAC7DD	User dir. : UKD0DDKD0	Report	Authorization review
PAC7EK	User dir. : UKD0EKKD0	Report	Validation reports
PAC7MD	User dir. : MUKD0.txt	Output	Rights' matching

Transaction formatting: PAF900

Code	Physical name	Type	Label
PAC7AR	Admin. Database - Base dir.: AR	Input	Administration Database data
PAC7AN	Admin. Database - Base dir.: AN	Input	Administration Database index
PAC7AE	System - Skel. dir.: AE	Input	Error labels
PACGGR	Admin. Base - Base dir.: AR	Input	Administration Database data
PACGGN	Admin. Base - Base dir.: AN	Input	Administration Database index
PACGGU	Admin. Base - Base dir.: GU	Input	Administration Database users
PAC7GY	Tmp dir.: WGY or WGZ	Input	Update transactions
PAC7MV	Tmp dir.: WMV	Output	Formatted transactions (should be able to contain all input transactions and the elementary cancel transactions generated by multiple cancel transactions) (length=170)
PAC7ME	Tmp dir.: WME	Output	Work file (length=372)
PAC7MW	Tmp dir.: WMW	Output	Work file (length=170)
PAC7MX	Tmp dir.: WMX	Output	Work file (length=743)
PAC7MY	Tmp dir.: WMY	Output	Work file (length=743)

UKD1 - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'     -   PREPARATION BEFORE ACTIVATION KEY        -
'
' ------------------------------------------------------
'
<job id=UKD0>

<script language="VBScript">
Dim MyProc
MyProc = "UKD0"
</script>

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

```
<script language="VBScript">

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

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

Call Msg_Log (Array("1022" , "PTUKD0"))
'-----------------------------------
WshEnv("PAC7MB") = Fic_Input
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_ABASE & "\AN"
WshEnv("PAC7AR") = Rep_ABASE & "\AR"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
Call BvpEnv("PTUKD0","PAC7EK",Rep_USR & "\UKD0EKKD0.txt")
Call BvpEnv("PTUKD0","PAC7DD",Rep_USR & "\UKD0DDKD0.txt")
Call BvpEnv("PTUKD0","PAC7MD",RepT_USR & "\INUKD1.txt")
Call RunCmdLog ("BVPTUKD0")
Call Err_Cod(Return , 0 , "PTUKD0")


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


</script>
</job>
```

UKD1 - Introduction

Principle

This procedure has three functions, knowing that only one can be performed at a time:

- Activate a new Access Key,
- Assign use rights to Users,
- Assign use rights to Profiles.

Execution conditions

The Administration Database files must be closed to online use.

Printed output

This procedure outputs a report which lists the errors, as well as an update review.

Result

The procedure updates the Administration Database according to the requests specified in input.

UKD1 - User input

This procedure requires two input file:

The first (PAC7MB) contains the '*' line, where the user code and password are specified.

The user must be an Administrator.

The second (PAC7MD) contains the data generated by the UKD0 procedure or entered manually.

It must contain, in an exclusive way, a 'K'-type line followed by 'R'-type lines, or 'U'-type lines, or 'P'-type lines.

'K'-type line to activate the key

Pos.	Len.	Value	Meaning
1	1	'K'	Line code to activate the key
2	14		Identifier of the key to be activated (formatted as CCYYMMDDHHMMSS). This key must exist in the Administration Database as 'Archive', must not be a test key and must not be expired.

'R'-type lines to match the rights of the old and the new key.

Pos.	Len.	Value	Meaning
1	1	'R'	Line code for the matching of rigts
2	4		Rights in the old key
6	4		Rights in the new key

'U'-type lines to modify or assign use rights to Users

Position	Length	Value	Meaning
2	1	'U'	Line code to modify or assign use rights
3	8		User code
11	50	TYPEx (1)	TYPE (4 characters) is the type of use right to be assigned (maximum of 10 seats by line)

(1) The x character corresponds to the permanent assignment of the right ('1' if permanent, ' ' if not permanent).

The possible values of the type of use rights are:

Value	Meaning
CO1	Designer
IW1	Workbench developer
PA1	TUI developer
T1n	Traditional build facility Mono-target n
T2n	Traditional build facility Bi-target n
TM1	Traditional build facility Multi-target
E1n	eBusiness build facility Mono-target n
E2n	eBusiness build facility Bi-target n
EM1	eBusiness build facility Multi-target

'P'-type line to assign rights to Profiles

Pos.	Len.	Value	Meaning
1	1	'P'	Line code to assign rights to Profiles
2	20		Profile code
22	4		Type of rights to assign
26	1	'1'	Permanent right
		' '	Not permanent right

Note:

The rights assigned to a Profile are also assigned to all the Users who have this Profile, provided these rights are compatible with those already assigned to the User.

UKD1 - Description of Steps

Processing the new key and assigning rights to several users: PTUKD1

Code	Physical Name	Type	Name
PAC7MB		Input	User input
PAC7MD	User dir. : MUKD0.txt	Input	Rights' matching

Code	Physical Name	Type	Name
PAC7AR	Admin Database - DBase dir.: AR	Input	Administration Database data
PAC7AN	Admin Database - DBase dir.: AN	Input	Administration Database index
PAC7AE	System - Skel dir.: AE	Input	Error messages
PACGGR	Admin Database - DBase dir.: AR	Input	Administration Database data
PACGGN	Admin Database - DBase dir.: AN	Input	Administration Database index
PACGGU	Admin Database - DBase dir.: GU	Input	Administration Database users
PAC7DD	User dir.: UKD1DDKD1	Report	Authorization check
PAC7EK	User dir.: UKD1EKKD1	Report	Error messages
PAC7GY	Tmp dir.: WGY	Output	User parameter transactions (length=310)

Transactions formatting: PAF900

Code	Physical name	Type	Label
PAC7AR	Admin Database - DBase dir.: AR	Input	Administration Database data
PAC7AN	Admin Database - DBase dir : AN	Input	Administration Database index
PAC7AE	System - Skel. dir.: AE	Input	Error messages
PACGGR	Admin Database - DBase dir.: AR	Input	Administration Database data
PACGGN	Admin Database - DBase dir.: AN	Input	Administration Database index
PACGGU	Admin Database - User dir.: GU	Input	Administration Database users
PAC7GY	Tmp dir.: WGY	Input	Update transactions
PAC7MV	Tmp dir.: WMV	Output	Formatted transactions (must be able to contain all the input transactions and the elementary delete transactions generated by the multiple delete transactions) (length = 170)
PAC7ME	Tmp dir.: WME	Output	Work file (length=372)

Code	Physical name	Type	Label
PAC7MW	Tmp dir.: WMW	Output	Work file (length=170)
PAC7MX	Tmp dir.: WMX	Output	Work file (length=743)
PAC7MY	Tmp dir.: WMY	Output	Work file (length=743)

Update of the Administration Database: PACA15

Code	Physical name	Type	Label
PAC7AR	Admin. Database - Base dir.: AR	Output	Administration Database Data file
PAC7AN	Admin. Database - Base dir.: AN	Output	Administration Database Index file
PAC7AY	Admin. Database - Base dir.: AY	Output	Administration Database extension
PAC7AJ	Admin. Database - Base dir.: AJ	Output	Administration Database journal
PAC7AE	System - Skel. dir.: AE	Input	Error messages
PACGGN	Admin. Database - Base dir.: AN	Input	Administration Database Index file
PACGGR	Admin. Database - Base dir.: AR	Input	Administration Database Data file
PACGGY	Admin. Database - Base dir.: AY	Input	Administration Database Extension
PACGGU	Admin. Database - Base dir.: GU	Input	Administration Database users
PAC7DC	NUL		
PAC7ME	Tmp dir.: WME	Input	Work file
PAC7MV	Tmp dir.: WMV	Input	Update transactions
PAC7RB	User dir.: RBA15	Output	UPDT erroneous transactions (length=80)
PAC7RY	User dir.: RYA15	Output	UPDP erroneous transactions (length=310)
PAC7IE	User dir.: IEA15	Report	Update report (length=132)
PAC7IF	User dir.: IFA15	Report	List of erroneous transactions (length=132)

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

Return codes:

- 0 : OK without error
- 2 : warning
- 4 : critical error

UKD1 - Execution JCL

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'     -      KEY ACTIVATION                          -
'     -  OR  ATTRIBUTION  OF USER'S RIGHTS           -
'     -  OR  ATTRIBUTION  OF PROFIL'S RIGHTS         -
'
' ------------------------------------------------------
'
'            UPDATE OPTION :
'
'     IF OPT = 0, UPDATE WITH 'BVPUPGP' PROCEDURE
'     IF OPT NOT = 0, UPDATE WITH 'BVPUPGA' PROCEDURE
'        FILES MUST BE OPENED
'
' ------------------------------------------------------
'
<job id=UKD1>

<script language="VBScript">
Dim MyProc
MyProc = "UKD1"
</script>

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

<script language="VBScript">

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

Call Msg_Log (Array("1029" ))
'----------------------------
Call StateList (base, statusL)

If c_error = 1 then  Wscript.Quit (1) End If
If Not FSO.FileExists( RepT_USR & "\INUKD1.txt") Then
   Call CreateNULF(RepT_USR & "\INUKD1.txt")
End If

Call Msg_Log (Array("1022" , "PTUKD1"))
'-----------------------------------
WshEnv("PAC7MB") = Fic_Input
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_ABASE & "\AN"
WshEnv("PAC7AR") = Rep_ABASE & "\AR"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
```

```
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
Call BvpEnv("PTUKD1","PAC7EK",Rep_USR & "\UKD1EKKD1.txt")
Call BvpEnv("PTUKD1","PAC7DD",Rep_USR & "\UKD1DDKD1.txt")
Call BvpEnv("PTUKD1","PAC7GY",Rep_TMP & "\WGY.tmp")
Call BvpEnv("PTUKD1","PAC7MD",RepT_USR & "\INUKD1.txt")
Call RunCmdLog ("BVPTUKD1")
Call Err_Cod(Return , 0 , "PTUKD1")


Call Msg_Log (Array("1022" , "PAF900"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_ABASE & "\AN"
WshEnv("PAC7AR") = Rep_ABASE & "\AR"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
Call BvpEnv("PAF900","PAC7GY",Rep_TMP & "\WGY.tmp")
Call BvpEnv("PAF900","PAC7ME",Rep_TMP & "\WME.tmp")
Call BvpEnv("PAF900","PAC7MV",Rep_TMP & "\WMV.tmp")
Call BvpEnv("PAF900","PAC7MW",Rep_TMP & "\WMW.tmp")
Call BvpEnv("PAF900","PAC7MX",Rep_TMP & "\WMX.tmp")
CalL BvpEnv("PAF900","PAC7MY",Rep_TMP & "\WMY.tmp")
Call RunCmdlog ("BVPAF900")
Call Err_Cod(Return , 0 , "PAF900")

Call Msg_Log (Array("1022" , "PACA15"))
'--------------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AJ") = Rep_AJOURNAL & "\AJ"
WshEnv("PAC7AN") = Rep_ABASE & "\AN"
WshEnv("PAC7AR") = Rep_ABASE & "\AR"
WshEnv("PAC7AY") = Rep_ABASE & "\AY"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PACGGY") = Rep_ABASE & "\AY"
WshEnv("SEMLOCK") = Rep_BASE & "\LO"
WshEnv("SEMADMIN") = Rep_ABASE & "\LO"
Call BvpEnv("PACA15","PAC7IE",Rep_USR & "\UKD1IEA15.txt")
Call BvpEnv("PACA15","PAC7IF",Rep_USR & "\UKD1IFA15.txt")
Call BvpEnv("PACA15","PAC7ME",Rep_TMP & "\WME.tmp")
Call BvpEnv("PACA15","PAC7MV",Rep_TMP & "\WMV.tmp")
WshEnv("PAC7DC") = Rep_TMP & "\NULDC.tmp"
'PAC7DC not used, on default
WshEnv("PAC7RB") = Rep_TMP & "\NULRB.tmp"
'PAC7RB not used, on default
WshEnv("PAC7RY") = Rep_TMP & "\NULRY.tmp"
'PAC7RY not used, on default
Call RunCmdLog ("BVPACA15")
Call Err_Cod(Return , 0 , "PACA15")

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

```
Call DeleteFldr (Rep_TMP )

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


</script>
</job>
```

Database statistics

STAT - Introduction

This procedure creates, from the files output by the save procedure, a sequential file which can be used to calculate statistics.

Execution conditions

None.

Output result

This procedure outputs a sequential file whose records are 100 characters long and whose contents are:

- a record which identifies the Database

Pos.	Len.	Format/Value	Meaning
9	10	'BASDEF '	Constant
19	4	9(4)	Database code
23	4	9(4)	Current session
27	4	9(8)	Current date

- a global record:

Pos.	Len.	Format/Value	Meaning
9	10	'GLOBAL '	Constant
19	3	9(3)	Number of libraries
22	8	9(8)	Index: number of records
30	8	9(8)	Index: number of deleted records
38	8	9(8)	Data: number of records (gaps + data)
46	8	9(8)	Data: number of deleted records
54	8	9(8)	Data: number of 'gap' records
62	8	9(8)	Data: number of 'data' records

Pos.	Len.	Format/Value	Meaning
70	8	9(8)	Data: number of records in frozen sessions
78	8	9(8)	Data: number of records in current sessions
86	8	9(8)	Bulk: number of records

- One or more records per library which has lower-level libraries:

Pos.	Len.	Format/Value	Meaning
1	3	X(3)	Library code
9	6	'LIBINF'	Constant (BIBINF for a French Database)
16	2	X(2)	First record: ' ' Next records: 01, 02, 03, etc ...
19	81		Table containing 27 items (3-character long) which represent lower-level libraries

- One or more records per library which has higher-level libraries:

Pos.	Len.	Format/Value	Meaning
1	3	X(3)	Library code
9	6	'LIBSUP'	Constant (BIBSUP for a French Database)
16	2	X(2)	First record: ' ' Next records: 01, 02, 03, etc ...
19	81		Table containing 27 items (3-character long) which represent higher-level libraries

- One record per library/session/line type:

Pos.	Len.	Format/Value	Meaning
1	3	X(3)	Library code
4	4	9(4)	Session number
8	1	X(1)	Session type
9	10	X(10)	Line type
19	8	9(8)	Number of updated data items (1)
27	8	9(8)	Number of deleted data items (1)

These numbers represent the number of entities which were updated or deleted during a given session.

The line type corresponds to the PAF table code. The following codes have been added:

- $ttDSC/YttDSC: User Entity description, tt being the call type (each description type is not detailed),
- xxxLOCKS: locks and timestamps, xxx being the entity type: TXT, DEL,... (xxxBLOCAGE for a French Database),
- xxxKWD: keywords, xxx being the entity type: TXT, DEL, ... (xxxMCL for a French Database),
- LONGV3: Long data attached to comments (-GC), generation elements (-GG), generation options (-GO) and error messages lines (-GE),
- LONG4: Long data attached to the layout lines of Reports,
- LONGY3: Long data attached to the User Entity Definition,
- LONGY4: Long data attached to the User Entity Descriptions.

STAT - User Input

A '*' line with the user code and password.

STAT - Description of Steps

Formatting of the sequential file: PTUSTA

Code	Physical Name	Type	Name
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database index
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database data
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database users
PAC7AR	Database dir. : AR	Input	Development Database data
PAC7PC	Save dir. : PC	Input	Sequential image of the Development Database
PAC7PD	Save dir. : PCI	Input	If Dispatch option in the save procedure: Sequential image 2 of the network
PAC7PY	Save dir. : PCY	Input	If Dispatch option in the save procedure: Sequential image 3 of the network
PAC7MB	User input	Input	User transactions
PAC7ST	User dir. : ST	Output	Output file (length= 100)
PAC7DD	User dir. : STATDD	Report	Execution report

STAT - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'            GENERATE DATABASE STATISTICS
'
' ------------------------------------------------------
'
<job id=STAT>

<script language="VBScript">
Dim MyProc
MyProc = "STAT"
</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" , "PTUSTA"))
'-----------------------------------
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("PAC7AR") = Rep_BASE & "\AR"
Call BvpEnv("PTUSTA","PAC7DD",Rep_USR & "\STATDD.txt")
Call BvpEnv("PTUSTA","PAC7ST",Rep_USR & "\ST")
WshEnv("PAC7PC") = BVP_SvName & ""
WshEnv("PAC7PD") = BVP_SvName & "I"
WshEnv("PAC7PY") = BVP_SvName & "Y"
Call RunCmdLog ("BVPTUSTA")
WshVolEnv("RC") = Return
Call Err_Cod(Return , 0 , "PTUSTA")

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

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

</script>
</job>
```

Chapter 5. Analysis of Activity and Quality Control

Analysis 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.

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

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 the 'Keywords Meaning and filling modes' paragraph.

The Output line is required; the Page and Area lines are optional. 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'.

The parameters which define counting or calculations are called 'Calculations'.

The coding, meaning and compatibility of the parameters are described in the 'Parameters Definition and Comments' paragraph.

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: Meaning 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 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 columns or 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 / Entity 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 the batch input mode; the value 'T' corresponds to the on-line input mode ,the value 'C' corresponds to special on-line input mode related to the GP screen and the value 'I' corresponds to the input transactions resulting from VINS and the bulk data.

&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 : End 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 five-character field: a four-character session field and one-character session status field.

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 five-character session field: a four-character session number and a one-character session status. 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 (except logon and logoff transactions which are not taken into account).

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

Note: &INTR represents the total of input transactions and not the total of the other selected transaction types.

&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 time

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 time format.

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

&H2 : End time

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 time format.

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

&MIN : Minute-by-minute 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-hour 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 : Change number

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

Parameter	AREa	PAGes	OUTput	OUTput
&RETR				CALCULATION

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

CALCULATION : only used in the Activity Calculation Mode.

ACTI - User Input

A '*' line with user code and password.

The specific input needed for this procedure is described in the previous section, titled 'Query language'.

ACTI - Description of Steps

Extraction: PTU630

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database index
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database data
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database users
PAC7AR	Database dir. : AR	Output	Development Database data
PAC7AN	Database dir. : AN	Output	Development Database index
PAC7PJ	Save dir. : PJ	Input	Archived update transactions
PAC7MB	User input	Input	Update transactions
PAC7ST	Tmp dir. : WST	Output	Selected reports transactions (length=247)
PAC7DD	User dir. : ACTIDD630	Report	Batch procedure authorization option

Return codes:

- 0: OK
- 8: No authorization on batch procedures.
- 12: System error.

Printing of results: PTU640

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7ST	Tmp dir. : WST	Input	Transactions for selected reports
PAC7IV	User dir. : ACTIIV640	Report	Selected reports

ACTI - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'             - ACTIVITY ANALYSIS -
'
' ------------------------------------------------------
'
' 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 (C,M,D)
'    - DATE OF UPDATE
'    - SESSION NUMBER OF UPDATE
' ------------------------------------------------------
'
<job id=ACTI>

<script language="VBScript">
Dim MyProc
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"
Call BvpEnv("PTU630","PAC7ST",Rep_TMP & "\WST.tmp")
Call BvpEnv("PTU630","PAC7DD",Rep_USR & "\ACTIDD630.txt")
Call RunCmdLog ("BVPTU630")

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"
Call BvpEnv("PTU640","PAC7ST",Rep_TMP & "\WST.tmp")
Call BvpEnv("PTU640","PAC7IV",Rep_USR & "\ACTIIV640.txt")
Call RunCmdLog ("BVPTU640")

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 purchaser site,
- The Quality rule extraction component, customized by the purchaser site.

The components supplied upon installation are:

- 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).
- 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.)

Analysis

PQCA - Introduction

The PQCA procedure analyzes the quality of the applications, according to either standard rules or user-defined rules.

Characteristics

The procedure invokes a unique program (BVPACQ), which is a monitor which calls the various programs used by the procedure.

All the programs called by the monitor are therefore considered to be sub-programs of BVPACQ, 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 OLSD Screens (OSD)
- G for Pacbench C/S 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 BVPTUQ20 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. Files can remain open to on-line use.

Information:

The procedure can have two types of input:

1. The entity which is analyzed [resource id="PQCA"] or [WshEnv("BVP_Input")].
2. Selection parameters (optional) [resource id="PQCR"] or [WshEnv("BVP_Inpqc")].

If the PCQA procedure is executed and if the PQCE procedure has not been executed before, the MPQCE.txt file which contains the quality rules is initialized with the quality rules delivered in standard in the $PACDIR/SYS/SKEL directory.

This description with a double input is characteristic of this procedure and follows the instructions, for each one, described in the installation guide.

PQCA - User Input

See the 'PQC' Reference manual.

PQCA - Description of Steps

Quality analysis: PACQ

Code	Physical name	Type	Label
PAC7AR	Database dir. : AR	Input	Development Database data
PAC7AN	Database dir. : AN	Input	Development Database index
PAC7AY	Database dir. : AY	Input	Development Database extension data
PAC7AJ	Journal dir. : AJ	Input	Development Database Journal file

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGK	Admin Database - Base dir. : GK	Input Output	Generation rights
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database index
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database data
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database users
PAC7LB	Base dir. : LB	Input	Generation follow-up
PACQMF	User dir. : MPQCE	Input	Quality rules
PAC7SC	System - Skel dir. : SC	Input	Batch generation language skeleton
PAC7SG	System - Skel dir. : SG	Input	OLSD generation skeleton
PAC7SN	System - Skel dir. : SN	Input	Client/server meta entity skeleton
PAC7SS	System - Skel dir. : SS	Input	Map skeleton
PAC7ME	User input	Input	Entities to be analyzed
PACQMC	User dir. : PQCR	Input	Selection parameter input
PAC7IA	User dir. : PQCAIA	Report	PACQ execution output
PAC7ID	User dir. : PQCAID	Report	Documentation
PACQIB	User dir. : PQCAIB	Report	Selection parameter check
PACQIE	User dir. : PQCAIE	Report	Results by entity type
PACQIF	User dir. : PQCAIF	Report	Results by entity
PACQIG	User dir. : PQCAIG	Report	List of identifiers exceeding the identifiers limits
PAC7GB	User dir. : PQCAGB	Output	DBD generated Cobol and/or others generated Cobol files
PAC7GE	User dir. : PQCAGE	Output	OLSD generated Cobol
PAC7GG	User dir. : PQCAGG	Output	Pacbench C/S generated Cobol
PAC7GP	User dir. : PQCAGP	Output	Batch language generated Cobol
PAC7GV	User dir. : PQCAGV	Output	PDM generated Cobol

Code	Physical name	Type	Label
.....			Other files mentioned in the procedure are temporary files used in the programs flows.

After the execution, the generated streams are concatenated in the PQCAGB file.

PQCA - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'         - PACBENCH QUALITY CONTROL -
'
' ------------------------------------------------------
'
' THE PQCA PROCEDURE CARRIES OUT AN ANALYSIS OF THE
' QUALITY OF THE APPLICATIONS, ACCORDING TO EITHER
' STANDARD RULES OR USER-DEFINED RULES.
'
' ------------------------------------------------------
'
<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

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

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

If Not FSO.FileExists( RepT_USR & "\MPQCE.txt") Then
   Call CreateNULF(RepT_USR & "\MPQCE.txt")
   If FSO.FileExists( RepT_USR & "\MPQCE.txt") Then
     Call CopFil (Rep_SKEL & "\BVPQCR" & CodLang , _
                  RepT_USR & "\MPQCE.txt")
   End If
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("PAC7AJ") = Rep_JOURNAL & "\AJ"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7LB") = Rep_BASE & "\LB"
WshEnv("PACGGK") = Rep_ABASE & "\GK"

WshEnv("PACQMC") =  BVP_PQCR
Call BvpEnv("PACQ","PACQMF",RepT_USR & "\MPQCE.txt")

Call BvpEnv("PACQ","PAC7EE",Rep_TMP & "\WEE.tmp")
Call BvpEnv("PACQ","PAC7EG",Rep_TMP & "\WEG.tmp")
Call BvpEnv("PACQ","PAC7EP",Rep_TMP & "\WEP.tmp")
Call BvpEnv("PACQ","PAC7EV",Rep_TMP & "\WEV.tmp")
Call BvpEnv("PACQ","PAC7GB",Rep_USR & "\PQCAGB.txt")
Call BvpEnv("PACQ","PAC7GE",Rep_USR & "\PQCAGE.txt")
Call BvpEnv("PACQ","PAC7GG",Rep_USR & "\PQCAGG.txt")
Call BvpEnv("PACQ","PAC7GP",Rep_USR & "\PQCAGP.txt")
Call BvpEnv("PACQ","PAC7GV",Rep_USR & "\PQCAGV.txt")
Call BvpEnv("PACQ","PAC7IA",Rep_USR & "\PQCAIA.txt")
Call BvpEnv("PACQ","PACQIB",Rep_USR & "\PQCAIB.txt")
Call BvpEnv("PACQ","PAC7ID",Rep_USR & "\PQCAID.txt")
Call BvpEnv("PACQ","PACQIE",Rep_USR & "\PQCAIE.txt")
Call BvpEnv("PACQ","PACQIF",Rep_USR & "\PQCAIF.txt")
Call BvpEnv("PACQ","PACQIG",Rep_USR & "\PQCAIG.txt")
Call BvpEnv("PACQ","PAC7JG",Rep_TMP & "\WJG.tmp")
Call BvpEnv("PACQ","PAC7KD",Rep_TMP & "\WKD.tmp")
Call BvpEnv("PACQ","PAC7KE",Rep_TMP & "\WKE.tmp")
Call BvpEnv("PACQ","PAC7KF",Rep_TMP & "\WKF.tmp")
Call BvpEnv("PACQ","PAC7KG",Rep_TMP & "\WKG.tmp")
Call BvpEnv("PACQ","PAC7KP",Rep_TMP & "\WKP.tmp")
Call BvpEnv("PACQ","PAC7KS",Rep_TMP & "\WKS.tmp")
Call BvpEnv("PACQ","PAC7KU",Rep_TMP & "\WKU.tmp")
Call BvpEnv("PACQ","PAC7KV",Rep_TMP & "\WKV.tmp")

WshEnv("PAC7ME") = Fic_Input

Call BvpEnv("PACQ","PAC7MG",Rep_TMP & "\WMG.tmp")
```

```
Call BvpEnv("PACQ","PAC7MV",Rep_TMP & "\WMV.tmp")
Call BvpEnv("PACQ","PACQMJ",Rep_TMP & "\WMJ.tmp")
Call BvpEnv("PACQ","PACQMK",Rep_TMP & "\WMK.tmp")
Call BvpEnv("PACQ","PACQMM",Rep_TMP & "\WMM.tmp")
Call BvpEnv("PACQ","PACQMN",Rep_TMP & "\WMN.tmp")

If Not FSO.FileExists( Rep_TMP & "\WMO.tmp") Then
   Call CreateNULF(Rep_TMP & "\WMO.tmp")
End If

Call BvpEnv("PACQ","PACQMO",Rep_TMP & "\WMO.tmp")
Call BvpEnv("PACQ","PACQMZ",Rep_TMP & "\WMZ.tmp")
WshEnv("PAC7SC") = Rep_SKEL & "\SC" & CodLang
WshEnv("PAC7SG") = Rep_SKEL & "\SG" & CodLang
WshEnv("PAC7SN") = Rep_SKEL & "\SN" & CodLang
WshEnv("PAC7SS") = Rep_SKEL & "\SS" & CodLang
Call BvpEnv("PACQ","PAC7W1",Rep_TMP & "\WW1.tmp")
Call BvpEnv("PACQ","PAC7W2",Rep_TMP & "\WW2.tmp")
Call BvpEnv("PACQ","PAC7W3",Rep_TMP & "\WW3.tmp")
Call BvpEnv("PACQ","PAC7W4",Rep_TMP & "\WW4.tmp")

Call RunCmdLog ("BVPACQ")


If Return < 10  then
Call Msg_Log (Array("1062"))
End if
If Return = 10  then
Call Msg_Log (Array("1063"))
End if
If Return = 11  then
Call Msg_Log (Array("1065"))
End if
If Return > 11  then
Call Msg_Log (Array("1064"))
End if
Call Err_Cod(Return , 11 , "PACQ")

If Return < 12 then
'Information : Normal End
'-----------------------------------------
Return = 0
WshVolEnv("RC") = Return

 If Not FSO.FileExists(WshEnv("PAC7GB")) Then
  Set LogPqc = FSO.CreateTextFile(WshEnv("PAC7GB") , TRUE)
  LogPqc.Close
 End if
 GB = WshEnv("PAC7GB")

 'Write For Appending on PQCAGB.txt
 'COPY  GB + G. ==> GB
  Call CopMFil(GB , WshEnv("PAC7GE") ,GB )
  Call DelFile (WshEnv("PAC7GE"))
  Call CopMFil(GB , WshEnv("PAC7GG") ,GB )
```

```
  Call DelFile (WshEnv("PAC7GG"))
  Call CopMFil(GB , WshEnv("PAC7GP") ,GB )
  Call DelFile (WshEnv("PAC7GP"))
  Call CopMFil(GB , WshEnv("PAC7GV") ,GB )
  Call DelFile (WshEnv("PAC7GV"))


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


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


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


</script>
</job>
```

Extraction of Quality Rules

PQCE - Introduction

The PQCE procedure performs the extraction of the quality rules created by the user in his/her Database via the user entity supplied.

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 - Input / Processing / 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:

Position	Length	Value	Meaning
2	1	'*'	Line code
3	8	uuuuuuuu	User code
11	8	pppppppp	User password
19	3	bbb	Library code
22	4	nnnn	Session number (Blank=current session)
26	1	'T'	Session status if Test session
28	1		not used
29	4	'EXUE'	Extractor code

For further details, see the chapter 'Manager's Utilities' (PACX: Extractions) in this manual.

One command line:

Pos.	Len.	Value	Meaning
2	4	'W1EX'	Line code
6	1	'Y'	UE extraction identifier
7	1		Library selection code:
		'U'	Selected library
		'C'	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 which contains the 'compiled' customized quality rules, which can be processed by the PQCA procedure.

Printed report

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	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7AN	Database dir. : AN	Input	Development Database Index file
PAC7AR	Database dir. : AR	Input	Development Database Data file
PAC7AY	Database dir. : AY	Input	Development Database Extension Data
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database Index file
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database Data file
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database Users
PAC7PJ	NUL	Input	Archived transactions
PAC7MB	User input	Input	User input
PAC7BM	Tmp dir.: WBM	Input/Output	User input
PAC7MM	Tmp dir.: WMM	Input/Output	EXPU work file
PAC7MJ	NUL	Input/Output	EXPJ work file
PAC7TE	NUL	Input/Output	RMEN work file
PAC7RE	NUL	Input/Output	RMEN work file
PAC7RM	NUL	Input/Output	RMEN work file
PAC7WD	Tmp dir.: WWD	Input/Output	Extracted transactions
SYSEXT	Tmp dir.: WSY	Input/Output	Work file (indexed)
PAC7MV	NUL	Output	Extracted transactions for UPDT
PAC7MR	NUL	Output	Extracted transactions for REOR (EXPU)
PAC7MX	NUL	Output	Non extracted entities (PACX)
PAC7GY	NUL	Output	Extracted transactions for UPDP

Code	Physical name	Type	Label
PAC7TD	NUL	Output	Extracted transactions for CPSN
PAC7UE	Tmp dir.: WUE	Output	Extracted transactions for EXUE
PAC7IA	User dir.: PQCEIA	Report	General printing of program chains
PAC7DD	User dir.: PQCEDD	Report	Printing of abends on input transactions
PAC7EE	User dir.: PQCEEE	Report	Extraction reports
PAC7EP	User dir.: PQCEEP	Report	Extraction reports
PAC7EQ	User dir.: PQCEEQ	Report	Extraction reports
PAC7EZ	User dir.: PQCEEZ	Report	Extraction reports

Return codes:

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

Compilation of Quality rules: PTUQ10

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

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database index
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database data
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database users
PAC7AR	Database dir. : AR	Input	Development Database data
PACQMI	User dir. : MPQCE	Output	'Compiled' quality rules (length=80)
PAC7MB	User input	Input	User input
PACQMC	Tmp dir. : WUE	Input	User input occurrences

Code	Physical name	Type	Label
PACQML	Tmp dir. : WML	Output	Preparation for printing
PACQIC	User dir. : PQCEICQ10	Report	Rule-validity report
PAC7DD	User dir. : PQCEDDQ10	Report	Batch procedure authorization option

Printing of Quality rules: PTUQ15

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACQML	Tmp dir. : WML	Input	Preparation for printing
PACQII	User dir. : PQCEIIQ15	Report	List of quality factors and criteria and description by indicator
PACQIJ	User dir. : PQCEIJQ15	Report	Dictionary of quality rules

PQCE - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'          - PACBENCH QUALITY CONTROL EXTRACTION -
'
' ------------------------------------------------------
'
' FORMAT OF TRANSACTIONS AT INPUT :
' .. A USER AND LIBRARY LINE
' .. A COMMAND LINE PER ENTITY TO BE EXTRACTED
'    COL  2-6  : "W1EXY"
'    COL  7    : SELECTION CODE OF THE LIBRARY
'                "U"(LIBRARY ONLY)
'                "C"(LIBRARY AND HIGHER LEVEL LIBRAIRIES)
'    COL  8-9  : TYPE CODE OF THE USER ENTITY (2 CHAR.)
'
' ------------------------------------------------------
'
<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"
Call BvpEnv("PACX","PAC7IA",Rep_USR & "\PQCEIA.txt")
Call BvpEnv("PACX","PAC7DD",Rep_USR & "\PQCEDD.txt")
Call BvpEnv("PACX","PAC7EE",Rep_USR & "\PQCEEE.txt")
Call BvpEnv("PACX","PAC7EZ",Rep_USR & "\PQCEEZ.txt")
Call BvpEnv("PACX","PAC7EP",Rep_USR & "\PQCEEP.txt")
Call BvpEnv("PACX","PAC7EQ",Rep_USR & "\PQCEEQ.txt")
WshEnv("PAC7GY") = Rep_TMP & "\NULGY.tmp"
'PAC7GY not used, on default
Call BvpEnv("PACX","PAC7BM",Rep_TMP & "\WBM.tmp")

WshEnv("PAC7MB") = Fic_Input

Call BvpEnv("PACX","PAC7MM",Rep_TMP & "\WMM.tmp")
WshEnv("PAC7MV") =  Rep_TMP & "\NULMV.tmp"
'PAC7MV not used, on default
WshEnv("PAC7MR") =  Rep_TMP & "\NULMR.tmp"
'PAC7MR not used, on default
WshEnv("PAC7MX") =  Rep_TMP & "\NULMX.tmp"
'PAC7MX not used, on default
WshEnv("PAC7PJ") = Rep_TMP & "\NUL.tmp"
'PAC7PJ not used, on default
WshEnv("PAC7MJ") =  Rep_TMP & "\NULMJ.tmp"
'PAC7MJ not used, on default
WshEnv("PAC7TE") =  Rep_TMP & "\NULTE.tmp"
'PAC7TE not used, on default
WshEnv("PAC7RE") =  Rep_TMP & "\NULRE.tmp"
'PAC7RE not used, on default
WshEnv("PAC7RM") =  Rep_TMP & "\NULRM.tmp"
'PAC7RM not used, on default
Call BvpEnv("PACX","SYSEXT",Rep_TMP & "\WSY.tmp")
WshEnv("PAC7TD") =  Rep_TMP & "\NULTD.tmp"
'PAC7TD not used, on default
Call BvpEnv("PACX","PAC7UE",Rep_TMP & "\WUE.tmp")
Call BvpEnv("PACX","PAC7WD",Rep_TMP & "\WWD.tmp")
Call RunCmdLog ("BVPACX")

If Return = 4 Then
Call Msg_Log (Array("1030"))
End If
If Return = 8 Then
Call Msg_Log (Array("1057"))
End If
Call Err_Cod(Return , 0 , "PACX")
```

```
Call Msg_Log (Array("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"
Call BvpEnv("PTUQ10","PAC7DD",Rep_USR & "\PQCEDDQ10.txt")
Call BvpEnv("PTUQ10","PACQIC",Rep_USR & "\PQCEICQ10.txt")
WshEnv("PAC7MB") = Fic_Input
Call BvpEnv("PTUQ10","PACQMI",RepT_USR & "\MPQCE.txt")
Call BvpEnv("PTUQ10","PACQMC",Rep_TMP & "\WUE.tmp")
Call BvpEnv("PTUQ10","PACQML",Rep_TMP & "\WML.tmp")
Call RunCmdLog ("BVPTUQ10")
Call Err_Cod(Return , 0 , "PTUQ10")

Call Msg_Log (Array("1022" , "PTUQ15"))
'-------------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
Call BvpEnv("PTUQ15","PACQII",Rep_USR & "\PQCEIIQ15.txt")
Call BvpEnv("PTUQ15","PACQIJ",Rep_USR & "\PQCEIJQ15.txt")
Call BvpEnv("PTUQ15","PACQML",Rep_TMP & "\WML.tmp")
Call RunCmdLog ("BVPTUQ15")
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

SCM Tools Interface

Introduction

The SCM module (Support Configuration Management) is the VA Pac interface for configuration management tools. These tools can be third-party tools or the VA Pac configuration management tool.

The SCM module is used in both online and batch modes.

This chapter presents the SCM browser, accessible via Administrator or Developer workbench, and supplies the detailed documentation of the batch procedures, specific to SCM.

The detailed use of SCM in Administrator or Developer workbench is documented in the online help, accessible via the 'Help' menu, 'Help Home Page' choice, or via the F1 key on any focusable field or interface section.

NOTE: In the online help, 'instance' replaces 'entity', which is used throughout this chapter.

The SCM module is optional. 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 the entities generated from a Database and put into production.
   - Give the user information related 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.
   - Provide project follow-up to development teams in relation to generated entities.

2. INTERFACE WITH A THIRD-PARTY CONFIGURATION MANAGEMENT UTILITY

Moreover, the module is used to:

- Generate and automatically import the generated entities into a third-party configuration management tool with the appropriate parameters.
- Know, in the repository, the contexts of the configuration management tool, in which the generated entities are stored.
- Consult the last actions performed on these entities in the configuration management tool.
- Ensure the consistency between the production turnover data, stored in the Development Database, and the entities generated and managed by the configuration management tool. But a file extracted from the product must be shipped so as to compare it with another extracted file from the Development Database.

Definitions

1. SCM Environment

An application life cycle consists of various steps: development, tests, quality control, production, etc.

Each step can be defined as an SCM Environment. A default SCM Environment, identified by '*' characters, can be defined. Its purpose is to store the generated entities data which does not belong to any predefined SCM Environment.

An SCM Environment can be associated with a physical or logical context of a configuration management tool. This context is defined with the context parameters which are specified in the Environment definition.

An SCM Environment can consist of several Applications.

A default Application, identified by '*' characters, stores the generated entities which do not belong to any particular Application.

An SCM Application can consist of several sets of generated entities, each set corresponding to one entity type only.

The entity types processed by SCM are limited to the following list:

- Batch program (P entity),
- Dialogue/Screen (O entity),
- DBD description (B entity),
- COPY clauses (D entity).

as well as the eBusiness entities:

- Application,
- Folder,
- IT server,
- Communication monitor,
- Elementary component.

2. Identity of the generated entity

The VA Pac generated entity is identified by the information defined in the PACBASE-CONSTANTS variable, in the generated program. This Identity can be obtained upon generation with the 'Pacbase Constants' parameter specified in the Optional Command Lines Set.

You can always find the generated entity Identity by analyzing the source of this entity.

Important Note :

Some SCM procedures, in particular the inter-environments integrity control procedure, need the identification of the generated entity which is stored in the configuration management tool to compare the data saved in this tool with the data stored in the Development Database.

For this reason, it is recommended to manage the generated entity and its Identity as well to use all the SCM functions.

Concerning Endevor, SCM manages the generated entity Identity as a complementary object called INFOPAC.

NOTE: For a complete documentation on the ENDEVOR Interface, refer to the "VisualAge Pacbase/ENDEVOR Interface" Reference Manual for IBM MVS CICS or IBM MVS IMS.

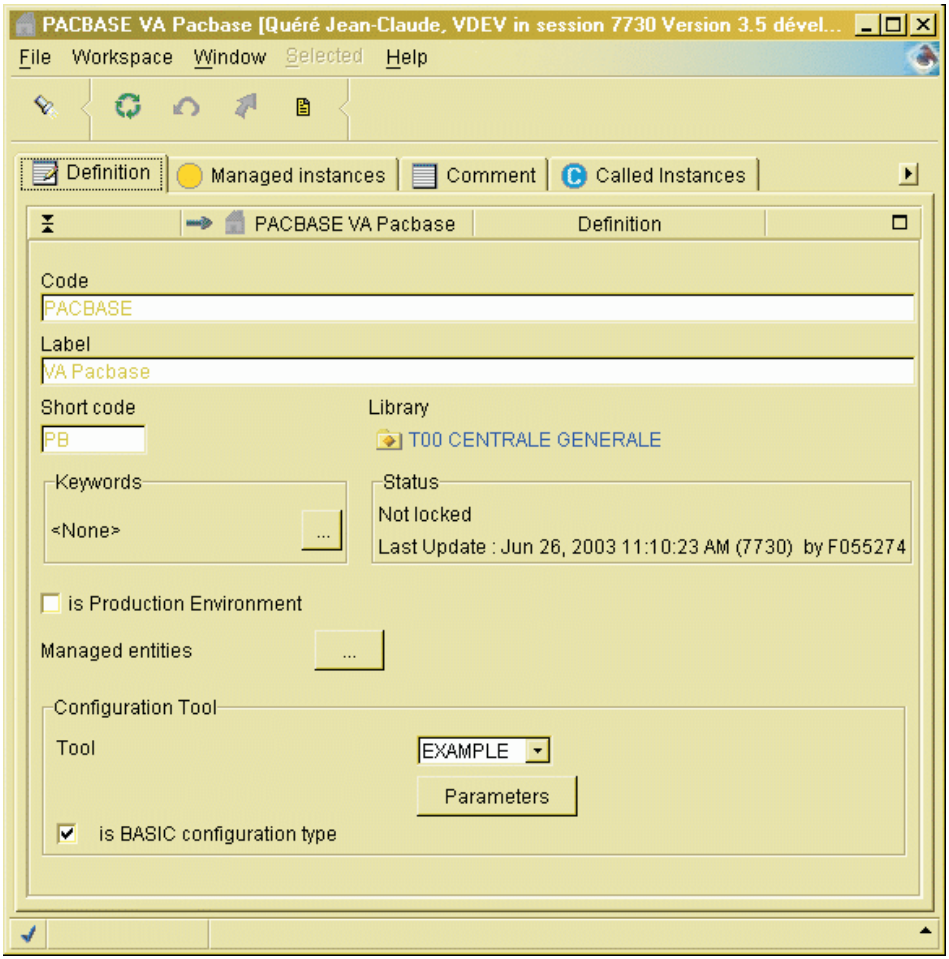
SCM Environment

The SCM Environment is a set of attributes and parameters which are optional. They are used to specify, for example, the contexts of the configuration management tools and so automate the generated entity import in these tools.

The values of these elements are used to configure the Optional Command Lines Set (OCLS) at generation time.

SCM Environment attributes

The SCM Environment attributes are defined in the 'Definition' tab of the 'Configuration Environments' browser.



These attributes are:

- Code of the Environment.

  It is unique and cannot exceed 30 characters.
- Label of the Environment.
- Short code of the Environment.

It is unique and is used in the FLx command of the print/generation command lines upon the entity generation, or in the Library options lines to specify the SCM Environment choice, either required of optional for a given entity type.

- Production Environment.

  Checking the box indicates that the defined Environment is the Production Environment. It triggers the Database freeze if entities to put into production exist in current session.

- Managed entities

  This attribute limits the SCM Environment use to some VA Pacbase entity types. At generation time, if the type of the generated entity is not permitted, this entity is not processed.

- Code of the configuration management tool.

  The default tool code is 'BASIC'. Other tool codes can be defined by adding the new tools in the 'tools.txt' file. This file is stored (and must remain) in the ..\adworkbench\workstation directory. Each configuration management tool can have its own parameters with specific labels and its own management. These labels are defined locally, in a file whose name is the name of the tool followed by the "txt" extension. This file is also located in the ..\adworkbench\workstation directory.

  Administrator or Developer workbench can then correctly display the parameters labels in the 'parameters' tab.

- BASIC-type configuration.

  If the box is checked, the Environment management is BASIC, i.e. only the latest instance of the generated entity is kept, in the Library, all frozen sessions taken into account.

Among the attributes, only the Environment code, label and short code are required.

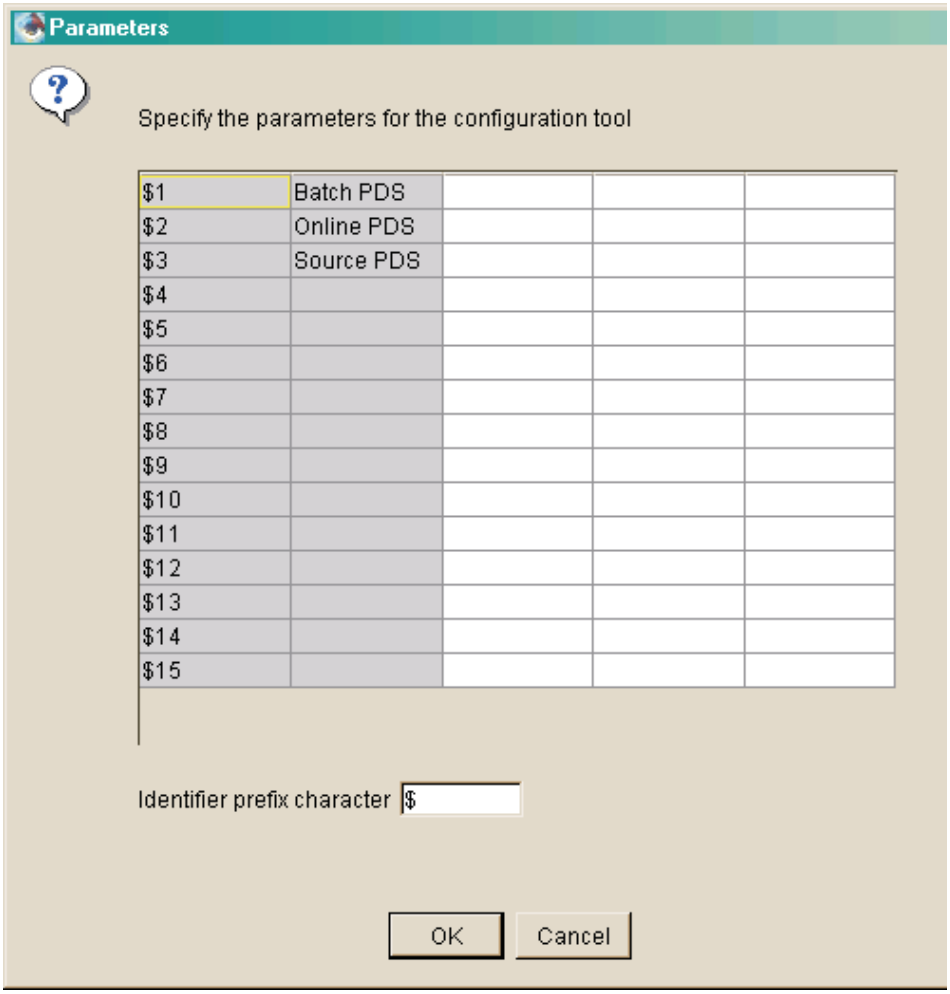
SCM Environment parameters

The parameters (number limited to 15) are values used by SCM :

- either to substitute the parameters entered in the Optional Command Lines Set of the generated entity at generation time. This, to import the generated entity in the configuration management tool.
- or to identify the physical or logical context of the configuration management tool where the generated entities are stored.

  To a SCM Environment corresponds only one context of the configuration management tool.

The parameters value can be assigned in the 'Parameters' window, opened from the 'Definition' tab of the 'Configuration Environments' browser.



Each parameter is constituted of the following attributes:

- Number, from 1 to 15 (column 1).
- Label (column 2).
- Value (column 3).
- Entity types (column 4).

  Here you can indicate whether the parameter is specific to an entity or generic for all entities.
- Rank (column 5).

  The parameter rank is used to sort the context parameters in case several parameters are needed to define a configuration management tool context. The values may be 1 to 9.

The generated entities which have a different type can be managed in different target contexts. So you can define several context parameters with the same rank, but with a different entity type.

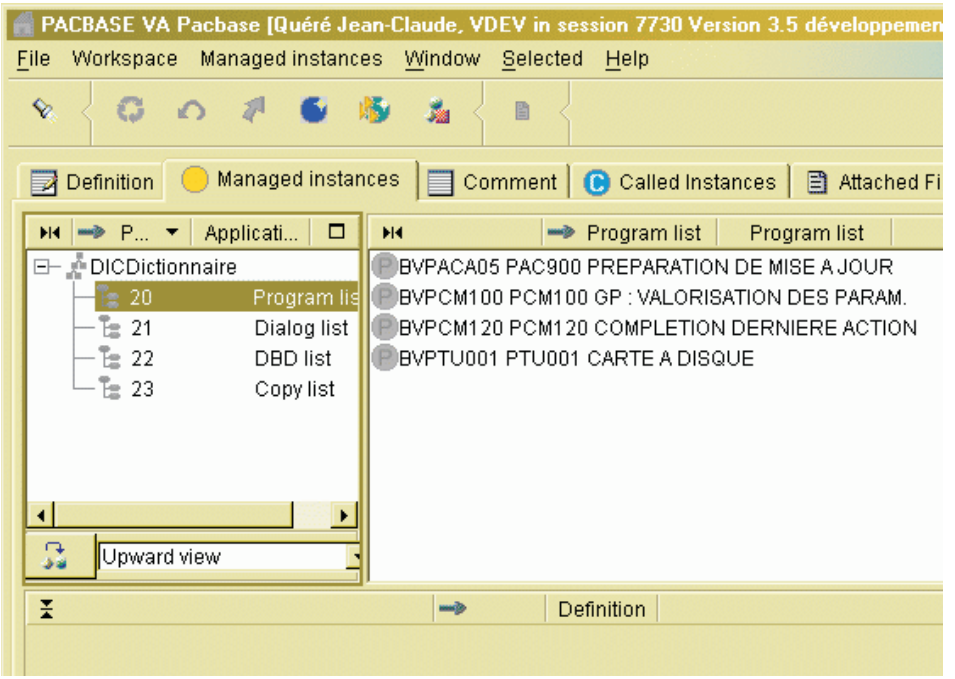
- Parameter identification prefix

  The value of the SCM Environment parameters can be used to substitute parameters which are indicated in the Optional Command Lines Set in order to possibly automate the the import of the generated entities in the configuration management tool upon the generation. In these lines, parameters are identified by a two-character sequence for the parameter identification followed by a number (1,2,...9,A,...,F).

  The default identification character is '$'.

The SCM Environment parameters value can be different according to the Library and the Session.

SCM Environment applications



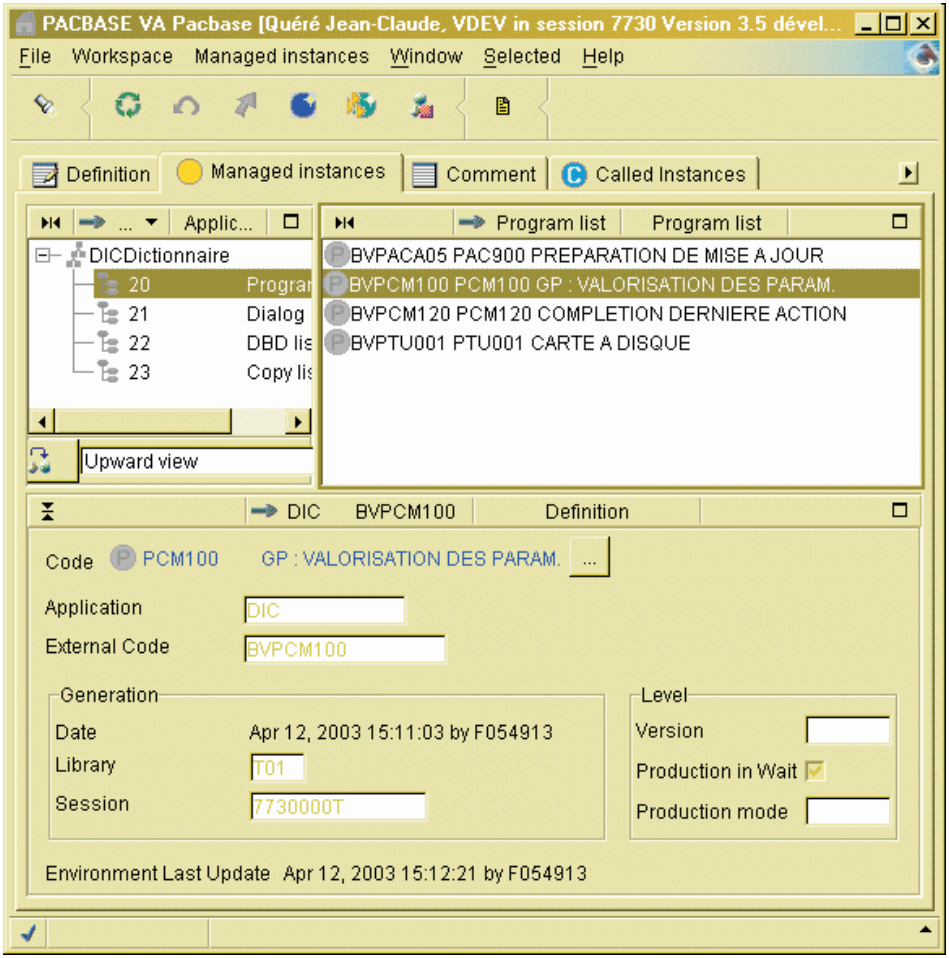
An SCM Environment can consist of one or more logical sets called 'Applications'.

Each Application groups VA Pacbase entities generated in a project, in an application.

The default Application, identified by '*' characters, stores the generated instances which do not belong to any specific Application, or whose Application code is not defined.

The definition of the Application is not required. If there is no defined Application code, the default Application is used.

SCM Environment managed entities



An Application consists of several sets of generated entities. Each set corresponds to a VA Pacbase entity type. You can specify the external name of the entities which belong to the Application before any generation.

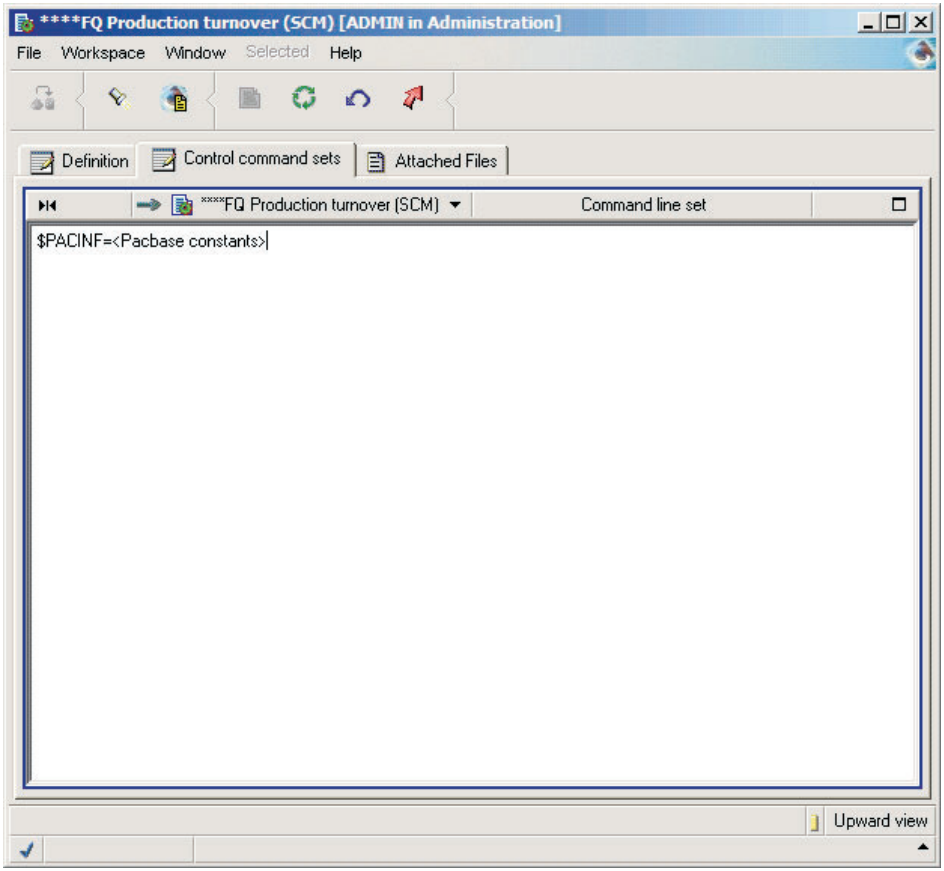
You can consult the Identity of the generated entities, and information about the last action performed on the entity.

A specific attribute indicates the status of the generated entity. It can be 'Production in wait' if this attribute is set.

The 'Production mode' attribute indicates the last batch procedure undergone by the entity, i.e. 'G' for GPRT, 'M' for HIPM and 'U' for SIPM.

The production turnover of a generated object in the current session can generate an automatic Database freeze.

Import Script



The import Script of the generated object in the configuration management tool must be defined in the Optional Command Lines Set.

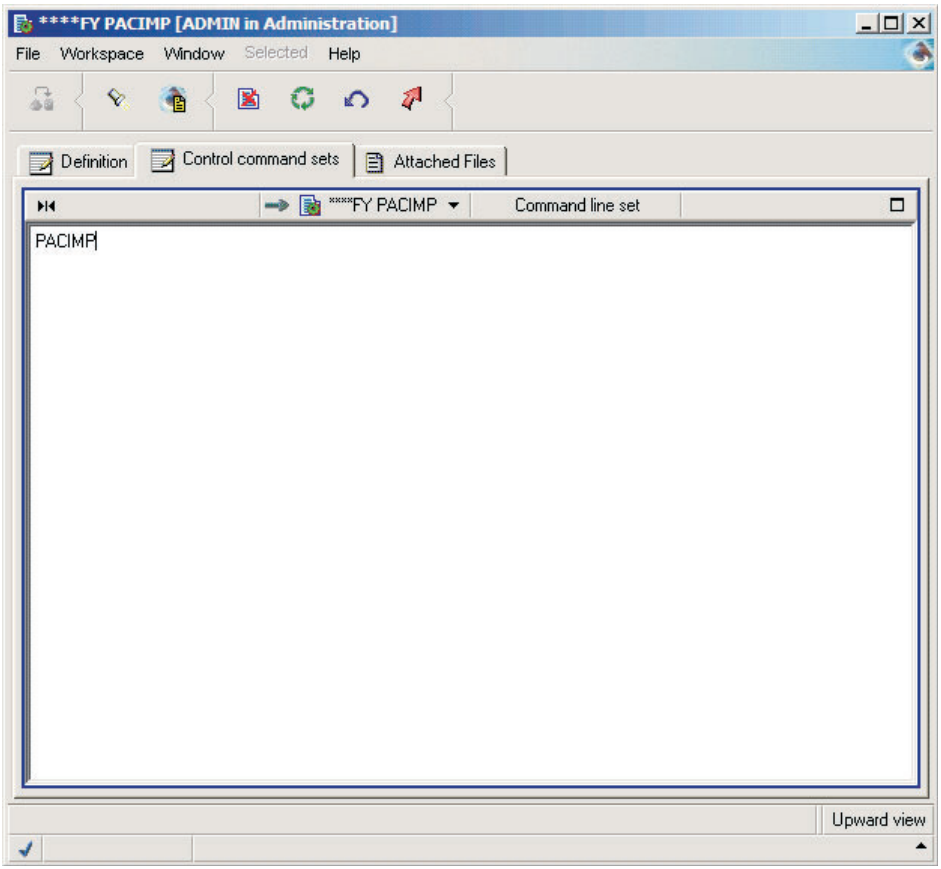
It can contain parameters which are valorized with the values specified in the SCM Environment parameters at generation time.

This lines set must be preceded by a separating line which contains the generated entity Identifier and whose structure is as follows:

$PACINF=<Pacbase constants>

This line allows SCM to locate the beginning of the set of lines specific to the generated entity, to search the corresponding parameters value to valorize these lines, and then to write the transaction corresponding to the entity generation action in the SCM QJ journal file.

The parameters interpretation begins with this separating line and the value of these parameters remains unchanged until the new separating line.



Once generated, the entity can be compiled/link-edited, and/or imported in a configuration management tool. If the operation is correctly ended, SCM can write a transaction which corresponds to the last action on the entity in QJ.

Therefore, an Optional Command Line Set (OCLS) must have the following content:

$PACIMP

Transaction lines are generated instead. They contain information related to the pacbase-constants of the generated entity. These lines are to be used as input to the post-generation procedure (GPPM), which must be executed after the compilation/link-edit and/or the import.

Choosing the Import SCM Environment

The import SCM Environment upon generation can be defined:

- by declaring, in the Library ″Option″ tab, the short code of the required or default SCM Environment, by entity type, possibly with the Application code:
- or by specifying the SCM Environment short code in the FLx command lines which are part of the generation command lines of the GPRT procedure, possibly with the corresponding Application code specified with the $APPLI parameter,
- or by defining first the generated entity in the SCM Environment entities list.

At generation time, the target import SCM Environment is chosen according to the following priority order:

1. Presence of the required SCM Environment definition in the Library ″Option″ tab, for the generated entity type, possibly with the corresponding Application code.
2. Presence of the SCM Environment short code in the FLx command lines of the generated entity, possibly with the Application code specified in the $APPLI parameter.
3. Declaration of the generated entity in the SCM Environment entities list.
4. Presence of the default SCM Environment definition in the Library ″Option″ tab for the generated entity type, possibly with the corresponding Application code.

Setting the Import Command Lines (OCLS)

In the Optional Command Lines Set used to generate the entity, you can specify parameters which will be substituted by the values of the parameters defined in the SCM Environment at generation time.

These parameters are identified by a two-characters sequence for the parameter identification prefix, followed by the corresponding parameter number (1,2,...9,A,...F).

Post-generation

GPPM - Introduction

Once the steps following the generation (compilation/linkedit and/or import in the configuration management tool) are executed, the purpose of the GPPM procedure is to write a transaction in the SCM QJ journal corresponding to the last action performed on the generated entity. This information completes the generated entity data in the SCM Environment instances list. This transaction is constituted from the transaction lines created by the generation procedure in place of the Optional Command Line after the generation of the generated entity which content is $PACIMP.

Execution conditions

The procedure must be executed after the generation and it is conditioned by a correct execution of the steps following the generation/import.

GPPM - Input / Processing / Results

The input transactions of the procedure are transactions created by the GPRT generation procedure, in place of the optional command line (OCLS) of the generated entity whose content is $PACIMP.

Printed reports

None.

Results

Once the procedure has been executed, a transaction which indicates the last action performed on the generated entity is written in the QJ journal file.

This transaction must be used as input to the UPPM procedure in order to be integrated into the Development Database.

Note: this transaction will also be taken into account when the generation procedure is executed again, with the option for the automatic application of the transactions written in QJ.

GPPM - Description of Steps

Input recognition: PTU001

Post-generation processing: PCM120

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7AN	Database dir. : AN	Input	Development Database index
PAC7AR	Database dir. : AR	Input	Development Database data
PAC7AY	Database dir. : AY	Input	Administration Database bulk data
PAC7IN	User input	Input	Generated transactions of the GPRT procedure
PAC7QJ	Admin Base - Journal dir. : QJ	Input/Output	SCM journal file

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'                     - POST-GENERATION -
'
' ------------------------------------------------------
'

<job id=GPPM>

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

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

<script language="VBScript">

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

'Input File issued from GPRT
'with $PACIMP defined on Command Lines(After)

Call Msg_Log (Array("1022" , "PCM120"))
'-------------------------------------
WshEnv("PAC7IN") = Fic_Input
If Not FSO.FileExists(WshEnv("PAC7IN")) Then
  Call Msg_Log (Array("1004" , "PAC7IN"))
  Msg = Nls_Lib
  EndJob (1)
  Wscript.Quit (1)
End If

WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7QJ") = Rep_AJOURNAL & "\QJ"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
```

```
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PAC7AY") = Rep_BASE & "\AY"
WshEnv("SEMLOCK") = Rep_BASE & "\LO"
WshEnv("SEMADMIN") = Rep_ABASE & "\LO"
Call RunCmdLog ("BVPCM120")
WshVolEnv("RC") = Return
Call Err_Cod(Return , 0 , "PCM120")


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

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

</script>
</job>
```

Database Automatic Freeze

HIPM - Introduction

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

Execution conditions

The Development Database must be closed to on-line use in order to be updated, except for hardware environments that support concurrent on-line and batch access.

Abnormal execution

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

HIPM - Input / Processing / 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:

Position	Length	Value	Meaning
2	2		Line code

Position	Length	Value	Meaning
		'X1'	if the entities 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 the number of the session 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 to be put into production are detected in the current session.

This line may be entered in order to:

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

If some entities are to be put into production, the 'X1' line is used to generate the Database freeze in the current session or in the session specified on the line, and then to generate the production turnover of these entities.

The 'X4' line is used to generate the Database freeze in the specified session whether or not some entities are to be put into production.

Moreover, if the 'X1' line is missing, the Database is automatically frozen in the current session if entities to be put into production (production in wait) have been detected. Several 'X4' lines can be created as input.

Printed reports

This procedure prints:

- a report,
- a list of the entities put into production, and the Database freeze (where applicable),
- a statistical report with the number of the entities to be put into production (production in wait) per library.

Results

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

HIPM - Description of Steps

Generation of production turnover transactions: PCM300

This step is used to explore the Development Database and to generate production turnover and Database freeze transactions.

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database index
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database data
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database users
PAC7AN	Database dir. : AN	Input	Development Database index
PAC7AR	Database dir. : AR	Input	Development Database data
PAC7AY	Database dir. : AY	Input	Development Database bulk data
PAC7MB	User input	Input	Users data .
PAC7TR	Tmp dir. : WTR	Output	Work file
PAC7SR	Tmp dir. : WSR	Output	Work file
PAC7UR	Tmp dir. : WUR	Output	Work file
PAC7IG	User dir. : HIPMIG300	Report	Production turnover output report
PAC7GY	User dir. : HIPMMY	Output	Production turnover transactions
PAC7DD	User dir. : HIPMDD300	Report	Batch procedures authorization option

Formatting of transactions: PAF900

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

Code	Physical name	Type	Label
PAC7GY	User dir. : HIPMMY	Input	Update transactions
PAC7MV	Tmp. dir.: WMV	Output	Formatting transactions (must be able to contain all the input transactions plus the elementary deletion transactions generated by the multiple deletion transactions) (length=170)
PAC7ME	Tmp. dir.: WME	Output	Work file (length=372)
PAC7MW	Tmp dir.: WMW	Output	Work file (length=170)
PAC7MX	Tmp dir.: WMX	Output	Work file (length=743)
PAC7MY	Tmp dir.: WMY	Output	Work file (length=743)

Update of the Development Database: PACA15

Code	Physical name	Type	Label
PAC7AR	Database dir.: AR	Output	Development Database Data file
PAC7AN	Database dir.: AN	Output	Development Database index
PAC7AY	Database dir.: AY	Output	Development Database extension
PAC7AJ	Journal dir.: AJ	Output	Development Database journal
PAC7AE	System - Skel. dir.: AE	Input	Error messages
PACGGN	Admin. DBase - Base dir.: AN	Input	Administration Database Index file
PACGGR	Admin. DBase - Base dir.: AR	Input	Administration Database Data file
PACGGY	Admin. DBase - Base dir.: AY	Input	Administration Database Extension
PACGGU	Admin. DBase - Base dir.: GU	Input	Administration Database users
PAC7DC	NUL	Input	DSMS file of Development Database Elements
PAC7ME	Tmp dir.: WME	Input	Work file
PAC7MV	Tmp dir.: WMV	Input	Update transactions
PAC7RB	NUL	Output	UPDT erroneous transactions (length=80)
PAC7RY	NUL	Output	UPDP erroneous transactions (length=310)
PAC7IE		Report	Update report (length=132)

Code	Physical name	Type	Label
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, no error
- 2: Warning
- 4: Error

HIPM - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
' AUTOMATIC SESSION FREEZE
' ------------------------------------------------------
'
' INPUT     : USER IDENTIFICATION
' COL 2     : "*"
' COL 3     : USER CODE
' COL 11    : PASSWORD
' ------------------------------------------------------
<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"
Call BvpEnv("PCM300","PAC7GY",Rep_USR & "\HIPMMY.txt")
Call BvpEnv("PCM300","PAC7DD",Rep_USR & "\HIPMDD300.txt")
Call BvpEnv("PCM300","PAC7IG",Rep_USR & "\HIPMIG300.txt")
WshEnv("PAC7MB") = Fic_Input
```

```
Call BvpEnv("PCM300","PAC7TR",Rep_TMP & "\WTR.tmp")
Call BvpEnv("PCM300","PAC7SR",Rep_TMP & "\WSR.tmp")
Call BvpEnv("PCM300","PAC7UR",Rep_TMP & "\WUR.tmp")
Call RunCmdLog ("BVPCM300")
Call Err_Cod(Return , 0 , "PCM300")


Call Msg_Log (Array("1022" , "PAF900"))
'-----------------------------------
Call BvpEnv("PAF900","PAC7GY",Rep_USR & "\HIPMMY.txt")
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
Call BvpEnv("PAF900","PAC7ME",Rep_TMP & "\WME.tmp")
Call BvpEnv("PAF900","PAC7MW",Rep_TMP & "\WMW.tmp")
Call BvpEnv("PAF900","PAC7MV",Rep_TMP & "\WMV.tmp")
Call BvpEnv("PAF900","PAC7MX",Rep_TMP & "\WMX.tmp")
Call BvpEnv("PAF900","PAC7MY",Rep_TMP & "\WMY.tmp")
Call RunCmdLog ("BVPAF900")
Call Err_Cod(Return , 0 , "PAF900")



Call Msg_Log (Array("1022" , "PACA15"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AJ") = Rep_JOURNAL & "\AJ"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PAC7AY") = Rep_BASE & "\AY"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGY") = Rep_ABASE & "\AY"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7DC") = Rep_TMP & "\WDC.tmp"
'PAC7DC not used, on default
Call BvpEnv("PACA15","PAC7IE",Rep_USR & "\HIPMIEA15.txt")
Call BvpEnv("PACA15","PAC7IF",Rep_USR & "\HIPMIFA15.txt")
WshEnv("SEMLOCK") = Rep_BASE & "\LO"
WshEnv("SEMADMIN") = Rep_ABASE & "\LO"
Call BvpEnv("PACA15","PAC7ME",Rep_TMP & "\WME.tmp")
Call BvpEnv("PACA15","PAC7MV",Rep_TMP & "\WMV.tmp")
WshEnv("PAC7RB") = Rep_TMP & "\WRB.tmp"
'PAC7RB not used, on default
WshEnv("PAC7RY") = Rep_TMP & "\WRY.tmp"
'PAC7RY not used, on default
Call RunCmdLog ("BVPACA15")
If Return = 2 Then
Call Msg_Log (Array("1061"))
End If
If Return = 4 Then
Call Msg_Log (Array("1060"))
```

```
End If
Call Err_Cod(Return , 4 , "PACA15")


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


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


</script>
</job>
```

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 is entered by the user.
- Transfer from one environment to another one:

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

Execution conditions

None.

The files can remain open.

Abnormal execution

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

SIPM - Input / Processing / Results

A '*' line with the user code and password, and information specific to the procedure.

Position	Length	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	ccyymmdd	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 SCM environment identification line: 'EG' (required):

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

An SCM source environment line: 'ES' (if transfer):

Position	Length	Value	Meaning
2	2	'ES'	Line code
7	30		Source environment
37	10		Source application

An entity identification line for each entity generation to be simulated : 'EU'

Position	Length	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)

Output report

This procedure prints a report.

Results

Once the procedure has been executed, production turnover simulation transactions are created in the QJ journal file.

These transactions must be input in the Development Database by executing the UPPM procedure.

SIPM - Description of Steps

Generation of simulation transactions: PCM320

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database index
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database data
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database users
PAC7AN	Database dir. : AN	Input	Development Database index
PAC7AR	Database dir. : AR	Input	Development Database data
PAC7AY	Database dir. : AY	Input	Administration Database bulk data
PAC7MB	User input	Input	User transactions
PAC7MT	Tmp dir. : WMT	Output	File to be used by a transfer utility
PAC7IE	User dir. : SIPMIE320	Report	Simulation output report
PAC7QJ	Admin Database - Journal dir. : QJ	Input/Output	Journal of SCM Tools Interface
PAC7DD	User dir. : SIPMDD320	Report	Batch procedures authorization option

SIPM - Execution Script

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

```
' ------------------------------------------------------
'
' INPUT     : USER IDENTIFICATION
' COL 2     : "*"
' COL 3     : USER CODE
' COL 11    : PASSWORD
' COL 19    : LIBRARY
' COL 22    : SESSION
' COL 25    : SESSION STATE
' ------------------------------------------------------
<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

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

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

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("PAC7QJ") = Rep_AJOURNAL & "\QJ"
Call BvpEnv("PCM320","PAC7DD",Rep_USR & "\SIPMDD320.txt")
Call BvpEnv("PCM320","PAC7IE",Rep_USR & "\SIPMIE320.txt")
WshEnv("PAC7MB") = Fic_Input
Call BvpEnv("PCM320","PAC7MT",Rep_TMP & "\WMT.tmp")
Call RunCmdLog ("BVPCM320")
Call Err_Cod(Return , 0 , "PCM320")



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 extracts, from the Development Database, the entities whose generation status is to be checked against the configuration management utility.

The extracted file will be compared to a file extracted from the utility.

The extraction can be limited to a Session, a Database, an Environment and an Application.

Execution conditions

None.

Abnormal execution

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

EXPM - Input / Processing / Results

A '*' line with user code, password and Library code.

Position	Length	Value	Meaning
2	1	'*'	Line code
3	8	uuuuuuuu	User code
11	8	pppppppp	Password
19	3	bbb	Library code
		'***'	if extraction from all Libraries
22	4	ssss	Session number (blank in current session)
		'****'	if extraction from all sessions
26	1		Session status (' ' or 'T')
		'*'	if extraction from all sessions

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

The line structure is as follows:

Position	Length	Value	Meaning
2	1	'S'	Line code
3	30		Selected environment
33	10		Selected application

Printed report

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	Physical name	Type	Label
PAC7MV	User input	Input	User transactions
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database index
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database data
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database users
PAC7AN	Database dir. : AN	Input	Development Database index
PAC7AR	Database dir. : AR	Input	Development Database data
PAC7AY	Database dir. : AY	Input	Development Database bulk data
PAC7ET	User dir. : EXPMET200	Report	Extraction output report

Code	Physical name	Type	Label
PAC7MS	Tmp dir. : WMS	Output	Extracted elements file
PAC7BM	Tmp dir. : WBM	Output	User assignment
PAC7DD	User dir. : EXPMDD200	Report	Batch procedures authorization option

Deletion of duplicate extracted elements: PCM202

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

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7AN	Database dir. : AN	Input	Development Database index
PAC7AR	Database dir. : AR	Input	Development Database data
PAC7ME	Tmp dir. : WMS	Input	Input extracted elements file
PAC7BM	Tmp dir. : WBM	Input	User assignment
PAC7MS	Tmp dir. : WMS2	Output	Output extracted elements file
PAC7EQ	User dir. : EXPMEQ202	Report	List of elements extracted twice

Extracted elements sort file: PCM205

This step sorts the extracted elements file in accordance with the criterion required for the comparison with the file provided by the configuration management utility.

Code	Physical name	Type	Label
PAC7ME	Tmp dir. : WMS2	Input	Input extracted elements file
PAC7MS	User dir. : MSEXPM	Output	Output extracted elements file

EXPM - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
' EXTRACTION
' ------------------------------------------------------
'
' INPUT     : USER IDENTIFICATION
' COL 2     : "*"
' COL 3     : USER CODE
```

```
' COL 11    : PASSWORD
' COL 19    : LIBRARY
' COL 22    : SESSION
' COL 26    : SESSION STATE
' ------------------------------------------------------
<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"
Call BvpEnv("PCM200","PAC7DD",Rep_USR & "\EXPMDD200.txt")
Call BvpEnv("PCM200","PAC7ET",Rep_USR & "\EXPMET200.txt")
WshEnv("PAC7MV") = Fic_Input
Call BvpEnv("PCM200","PAC7MS",Rep_TMP & "\WMS.tmp")
Call BvpEnv("PCM200","PAC7BM",Rep_TMP & "\WBM.tmp")
Call RunCmdLog ("BVPCM200")
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"
Call BvpEnv("PCM202","PAC7BM",Rep_TMP & "\WBM.tmp")
Call BvpEnv("PCM202","PAC7ME",Rep_TMP & "\WMS.tmp")
Call BvpEnv("PCM202","PAC7MS",Rep_TMP & "\WMS2.tmp")
Call BvpEnv("PCM202","PAC7EQ",Rep_USR & "\EXPMEQ202.txt")
Call RunCmdLog ("BVPCM202")
Call Err_Cod(Return , 0 , "PCM202")

Call Msg_Log (Array("1022" , "PCM205"))
'-------------------------------------
Call BvpEnv("PCM205","PAC7MS",RepT_USR & "\MSEXPM.txt")
Call BvpEnv("PCM205","PAC7ME",Rep_TMP & "\WMS2.tmp")
Call RunCmdLog ("BVPCM205")
Call Err_Cod(Return , 0 , "PCM205")
```

```
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 by the EXPM procedure, with a similar file extracted by the user from the configuration management utility.

The purpose is to generate transactions to update the Development Database. These transactions are saved in the SCM QJ journal file.

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

The user must start the UPPM procedure to integrate the corrections in the Repository.

Note: the transactions will also be taken into account when the generation is executed again, with the option for the automatic application of the transactions written in QJ.

Execution conditions

The EXPM procedure must be executed 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 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 - Input / Processing / Results

A '*' line with user code and password.

Printed report

This procedure prints:

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

Results

Once the procedure has been executed, the QJ file contains the Development Database update transactions to be used as input to the UBPM procedure.

CPPM - User File

To align the Development Database with the Configuration management utility used on-site, it is necessary to create a file which contains the data extracted from the utility so as to compare it with the file extracted from the Development Database via the EXPM procedure.

The length of the file should be 900 with the following structure:

Position	Length	Value	Meaning
1	35		Parameter on rank 1
36	35		Parameter on rank 2
71	35		Parameter on rank 3
106	35		Parameter on rank 4
141	35		Parameter on rank 5
176	35		Parameter on rank 6
211	35		Parameter on rank 7
246	35		Parameter on rank 8
281	35		Parameter on rank 9
316	35		Parameter on rank 10
351	35		Parameter on rank 11
386	35		Parameter on rank 12
421	35		Parameter on rank 13
456	35		Parameter on rank 14
491	35		Parameter on rank 15

Position	Length	Value	Meaning
526	30		SCM 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 (CCYY/MM/DD)
605	8		Generation time (HH:MM:SS)
613	8		User code
621	35		Group field
			Description of information areas of the third-party product concerning the last action on the object. (5 optional area):
796	20		Action label
816	10		Action date
826	8		Action hour
834	8		User code
842	5		Object version
847	54		VA Pac internal fields

The information of 'parameter 1' up to 'parameter 15' corresponds to the context parameters defined via the Definition tab of the Environment. The sorting order is taken into account.

The 'Entity type ' and the next information correspond to the values defined in the generated program, in the PACBASE-CONSTANTS (or CONTANTES-PACBASE).

CPPM - Description of Steps

Comparison processing: PCM210

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

Code	Physical name	Type	Label
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database index
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database data
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database users
PAC7AY	Database dir. : AY	Output	Development Database extension data
PAC7AN	Database dir. : AN	Input	Development Database index
PAC7AR	Database dir. : AR	Input	Development Database data
PAC7MB	User input	Input	User transactions
PAC7MP	User dir. : MSEXPM	Input	File extracted from the Development Database
PAC7MU	Tmp dir. : MUGCL	Input	File extracted from configuration management utility
PAC7EQ	User dir. : CPPMEQ210	Report	Output report of check
PAC7BM	Tmp dir. : WBM	Output	User assign
PAC7ME	Tmp dir. : WME	Output	File used to print comparison errors
PAC7MS	Tmp dir. : WMS	Output	File used to print update transactions
PAC7DD	User dir. : CPPMDD210	Report	Batch procedures authorization option

Print of update transactions: PCM220

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7AN	Database dir. : AN	Input	Development Database index
PAC7AR	Database dir. : AR	Input	Development Database data
PAC7EQ	User dir. : CPPMEQ220	Report	Comparison output report
PAC7BM	Tmp dir. : WBM	Input	File used to print the results of comparison
PAC7MS	Tmp dir. : WMS	Input	File used to print the results of comparison
PAC7QJ	Journal dir. : QJ	Input/Output	Journal of SCM Tools Interface

CPPM - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
' COMPARIZON
' ------------------------------------------------------
'
' INPUT     : USER IDENTIFICATION
' COL 2     : "*"
' COL 3     : USER CODE
' COL 11    : PASSWORD
' ------------------------------------------------------
<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"
Call BvpEnv("PCM210","PAC7DD",Rep_USR & "\CPPMDD210.txt")
Call BvpEnv("PCM210","PAC7EQ",Rep_USR & "\CPPMEQ210.txt")
WshEnv("PAC7MB") = Fic_Input
Call BvpEnv("PCM210","PAC7MP",RepT_USR & "\MSEXPM.txt")
Call BvpEnv("PCM210","PAC7MU",RepT_USR & "\MUGCL.txt")
Call BvpEnv("PCM210","PAC7ME",Rep_TMP & "\WME.tmp")
Call BvpEnv("PCM210","PAC7MS",Rep_TMP & "\WMS.tmp")
Call BvpEnv("PCM210","PAC7BM",Rep_TMP & "\WBM.tmp")
Call RunCmdLog ("BVPCM210")
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("PAC7QJ") = Rep_AJOURNAL & "\QJ"
Call BvpEnv("PCM220","PAC7MS",Rep_TMP & "\WMS.tmp")
Call BvpEnv("PCM220","PAC7EQ",Rep_USR & "\CPPMEQ220.txt")
```

```
Call BvpEnv("PCM210","PAC7BM",Rep_TMP & "\WBM.tmp")
Call RunCmdLog ("BVPCM220")
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 Environments/Elements

CHPM - Introduction

The CHPM procedure is used to check the consistency of the environments and elements present in the VA Pac Database. The procedure produces a report printout for erroneous environments and elements. The purpose of this check is to highlight the inconsistencies in the Development Database.

Execution condition

None.

Abnormal execution

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

CHPM - Input / Processing / Results

A '*' line with user code and password.

Printed report

This procedure prints a report which lists 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	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database index
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database data
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database users
PAC7AN	Database dir. : AN	Input	Development Database index
PAC7AR	Database dir. : AR	Input	Development Database data
PAC7AY	Database dir. : AY	Input	Development Database bulk data
PAC7MB	User input	Input	User transactions
PAC7MS	Tmp dir. : WMS	Output	File used to print a check report
PAC7MV	Tmp dir. : WMV	Output	Work file
PAC7DD	User dir. : CHPMDD400	Report	Batch procedures authorization option

Consistency check report printout: PCM410

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7EQ	User dir. : CHPMEQ410	Report	Output validation report
PAC7MS	Tmp dir. : WMS	Input	File used for the report printout
PAC7AN	Database dir. : AN	Input	Development Database index
PAC7AR	Database dir. : AR	Input	Development Database data

CHPM - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
' VALIDATION OF THE DEVELOPMENT DATABASE
' ------------------------------------------------------
'
' INPUT     : USER IDENTIFICATION
' COL 2     : "*"
' COL 3     : USER CODE
```

```
' COL 11    : PASSWORD
' -----------------------------------------------------
<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"
Call BvpEnv("PCM400","PAC7DD",Rep_USR & "\CHPMDD400.txt")
WshEnv("PAC7MB") = Fic_Input
Call BvpEnv("PCM400","PAC7MV",Rep_TMP & "\WMV.tmp")
Call BvpEnv("PCM400","PAC7MS",Rep_TMP & "\WMS.tmp")
Call RunCmdLog ("BVPCM400")
Call Err_Cod(Return , 0 , "PCM400")

Call Msg_Log (Array("1022" , "PCM410"))
'-------------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
Call BvpEnv("PCM410","PAC7MS",Rep_TMP & "\WMS.tmp")
Call BvpEnv("PCM410","PAC7EQ",Rep_USR & "\CHPMEQ410.txt")
Call RunCmdLog ("BVPCM410")
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>
```

Update

UPPM - Introduction

With the SCM option, the GPRT generation procedure writes transactions, structured as PAF tables, to the QJ journal file. This procedure updates the VA Pac Database with these transactions.

The processing starts from the first non-processed QJ transaction. It prepares the loading of the communication area with PAF-type transactions extracted from QJ before the call to PUF for the update. PUF function is called when a PAF file, the user code or the Database code is modified.

The returned errors, if any, are saved in the QJ file.

In case of system error related to the Database, the Database transactions are ignored during the processing.

During the archiving, the valid transactions are deleted from the QJ file while the erroneous transactions or the transactions which have not been processed are recycled to reconstitute the new QJ file for the next processing.

Execution condition

The QJ file must exist.

Since this procedure updates the Database, the AR, AN, AJ and AY files must be closed to online use (except on the platforms where batch/online concurrency is allowed).

UPPM - Input / Processing / Results

The UPPM procedure updates the VA Pac Databases from the QJ file which contains PAF Table formatted transactions written by the GPRT generation procedure upon generation.

Output reports

None.

UPPM - Description of Steps

Update : PCMPUF

This step updates the User Entities of the SCM Meta Entity in the VA Pac Database.

Code	Physical name	Type	Label
PAC7AE	System - Skel. dir.: AE	Input	Error messages
PACGGN	Admin. Database - Base dir.: AN	Input	Administration Database index
PACGGR	Admin. Database - Base dir.: AR	Input	Administration Database data
PACGGU	Admin. Database - Base dir.: GU	Input	Administration Database users
PAC7AN	Base dir.: AN	Input	Development Database index
PAC7AR	Base dir.: AR	Input	Development Database data
PAC7AY	Base dir.: AY	Input	Development Database unsorted data
PAC7IC	NUL	Input	DSMS Control data
PAC7QJ	Admin. Database - Journal dir.: QJ	Output	Standard bridge journal file

UPPM - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'                     - SCM UPDATE -
'
'
' ------------------------------------------------------

<job id=UPPM>

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

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

<script language="VBScript">

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

If Not FSO.FileExists( Rep_AJOURNAL & "\QJ") Then
   Call Msg_Log (Array("1053", "QJ"))
   Call Msg_Log (Array("1023"))
   Wscript.Quit (0)
End if

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

```
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7IC") = Rep_TMP & "\NUL.tmp"
'PAC7IC not used, on default
WshEnv("PAC7QJ") = Rep_AJOURNAL & "\QJ"
WshEnv("PAC7AJ") = Rep_JOURNAL & "\AJ"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PAC7AY") = Rep_BASE & "\AY"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("SEMLOCK") = Rep_BASE & "\LO"
WshEnv("SEMADMIN") = Rep_ABASE & "\LO"
Call RunCmdLog ("BVPCMPUF")
WshVolEnv("RC") = Return
Call Err_Cod(Return , 0 , "PCMPUF")


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

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

</script>
</job>
```

Transactions Archiving

ARPM - Introduction

The ARPM procedure archives the valid transactions already processed by the generation procedure or by the UPPM procedure and stored in the QJ journal file. It extracts the erroneous transactions, outputs the corresponding errors and recycles them again. It reconstitutes the new QJ journal file with the transactions not processed yet and with the recycled erroneous transactions.

Execution conditions

No current generation. The QJ file must be closed to on-line use.

Abnormal execution

If the last step (which reconstitutes the QJ journal file) abends, you just have to re-run this step. In all other cases, whichever the cause of the error is, you can run the procedure again as it is, once you have solved the problem.

ARPM - Input / Processing / Results

A '*' line with the user code and password.

Output reports

This procedure outputs a list of the errors previously detected by the update procedure.

ARPM - Description of Steps

Analysis and preparation: PCM500

This step validates the user input, analyzes the QJ journal file and prepares the archiving and the output of errors.

Code	Physical name	Type	Label
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database index
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database data
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database users
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7AR	Database dir.: AR	Input	Development Database data
PAC7MB	User input	Input	User transaction
PAC7QJ	Admin Database - Journal dir.: QJ	Input	SCM journal
PAC7XP	Tmp dir.: WXP	Output	Counters of the reconstituted journal
PAC7XQ	Tmp dir.: WXQ	Output	Transactions to be recycled
PAC7XR	Tmp dir.: WXR	Output	Erroneous transactions
PAC7XS	Tmp dir.: WXS	Output	Transactions already processed
PAC7XT	Tmp dir.: WXT	Output	Transactions for update
PAC7DD	User dir.: ARPMDD500	Report	Authorization control

Transactions archiingl: PCM510

This step archives the correct transactions already processed by the different updates.

Code	Physical name	Type	Label
PAC7AE	System - Skel dir.: AE	Input	Error messages
PAC7AR	Database dir. : AR	Input	Development Database data
PAC7XS	Tmp dir. : WXS	Input	Transactions already processed
PAC7RJ	Admin Database - Journal dir.: JQ	Input	Former archiving
PAC7JR	Admin Database - Journal dir.: JQ-new	Output	New archiving

Output of errors: PCM520

This step outputs the list of the errors detected by the update.

Code	Physical name	Type	Label
PAC7AE	System - Skel dir.: AE	Input	Error messages
PAC7AN	Base dir.: AN	Input	Development Database index
PAC7AR	Base dir.: AR	Input	Development Database data
PAC7XR	Tmp dir.: WXR	Input	Extracted erroneous transactions
PAC7ET	User dir.: ARPMET520	Report	List of errors

Preparation of the QJ file reconstitution: PCM550

This step prepares the reconstitution of the new QJ file with the transactions which have not been processed yet by the different updates as well as with the erroneous transactions.

Code	Physical name	Type	Label
PAC7XP	Tmp dir.: WXP	Input	Counters
PAC7XQ	Tmp dir.: WXQ	Input	Transactions to be processed again
PAC7JQ	Tmp dir.: WJQ	Output	New journal

Reconstitution of the QJ journal file: PCM560

This step reconstitutes the new QJ journal file, using as input the sequential file created by PCM550.

Code	Physical name	Type	Label
PAC7JQ	Tmp dir. : WJQ	Input	Sequential file

Code	Physical name	Type	Label
PAC7QJ	Admin base - Journal dir.: QJ-new	Output	New journal

ARPM - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'            - ARCHIVAL OF THE SCM JOURNAL -
'
'
' ------------------------------------------------------
' INPUT     : USER IDENTIFICATION
' COL 2     : "*"
' COL 3     : USER CODE
' COL 11    : PASSWORD
' ------------------------------------------------------

<job id=ARPM>

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

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

<script language="VBScript">

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

If Not FSO.FileExists(Rep_AJOURNAL & "\QJ") Then
   Call Msg_Log (Array("1053", "QJ"))
   Call Msg_Log (Array("1023"))
   Wscript.Quit (0)
End if
If Not FSO.FileExists(Rep_AJOURNAL & "\JQ") Then
   Set Fil_JQ = FSO.CreateTextFile(Rep_AJOURNAL & "\JQ", TRUE)
   Fil_JQ.Close
End if


Call Msg_Log (Array("1022" , "PCM500"))
'-------------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7QJ") = Rep_AJOURNAL & "\QJ"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
Call BvpEnv("PCM500","PAC7DD",Rep_USR & "\ARPMDD500.txt")
```

```
WshEnv("PAC7MB") = Fic_Input
Call BvpEnv("PCM500","PAC7XP",Rep_TMP & "\WXP.tmp")
Call BvpEnv("PCM500","PAC7XQ",Rep_TMP & "\WXQ.tmp")
Call BvpEnv("PCM500","PAC7XR",Rep_TMP & "\WXR.tmp")
Call BvpEnv("PCM500","PAC7XS",Rep_TMP & "\WXS.tmp")
Call BvpEnv("PCM500","PAC7XT",Rep_TMP & "\WXT.tmp")
Call RunCmdLog ("BVPCM500")
Call Err_Cod(Return , 0 , "PCM500")

Call Msg_Log (Array("1022" , "PCM510"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
Call BvpEnv("PCM510","PAC7XS",Rep_TMP & "\WXS.tmp")
WshEnv("PAC7RJ") = Rep_AJOURNAL & "\JQ"
WshEnv("PAC7JR") = Rep_AJOURNAL & "\JQ-new"
Call RunCmdLog ("BVPCM510")
Call Err_Cod(Return , 0 , "PCM510")

Call Msg_Log (Array("1022" , "PCM520"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
Call BvpEnv("PCM520","PAC7XR",Rep_TMP & "\WXR.tmp")
Call BvpEnv("PCM520","PAC7ET",Rep_USR & "\ARPMET520.txt")
Call RunCmdLog ("BVPCM520")
Call Err_Cod(Return , 0 , "PCM520")

Call Msg_Log (Array("1022" , "PCM550"))
'-----------------------------------
Call BvpEnv("PCM550","PAC7XP",Rep_TMP & "\WXP.tmp")
Call BvpEnv("PCM550","PAC7XQ",Rep_TMP & "\WXQ.tmp")
Call BvpEnv("PCM550","PAC7JQ",Rep_TMP & "\WJQ.tmp")
Call RunCmdLog ("BVPCM550")
Call Err_Cod(Return , 0 , "PCM550")

Call Msg_Log (Array("1022" , "PCM560"))
'-----------------------------------
WshEnv("PAC7QJ") = Rep_AJOURNAL & "\QJ-new"
Call BvpEnv("PCM560","PAC7JQ",Rep_TMP & "\WJQ.tmp")
Call RunCmdLog ("BVPCM560")
Call Err_Cod(Return , 0 , "PCM560")

Call Msg_Log (Array("1022" , "BACKUP"))
'-----------------------------------
Call Turnover(Rep_AJOURNAL& "\JQ")
Call Turnover(Rep_AJOURNAL& "\QJ")

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

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

```
Wscript.Quit (Return)

</script>
</job>
```

Purge

PUPM - Introduction

The purge procedure is used to analyze, in production intra-sessions, the data saved in the Repository related to the generated entities of BASIC-type SCM Environments. This procedure also generates deletion transactions of redundant data in order to keep the trace of the last generation only.

Execution conditions

As this procedure updates the network, the AR, AN, AJ and AY files must be closed to on-line use (except for hardware environments that support concurrent on-line and batch access).

Abnormal execution

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

PUPM - Input / Processing / Results

A '*' line with the user code and password of the Manager.

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

Printed output

This procedure prints a validity report for the SCM Environments encountered. Only the valid Environments will be processed subsequently.

Results

This procedure outputs a sequential file which contains the update transactions of the Development Database. This file must be used as input to the UPDP procedure to update the Repository.

PUPM - Description of Steps

Input recognition: PTU001

Extraction: PCM450

This step scans the Development Database and extracts the redundant data.

Code	Physical name	Type	Label
PAC7AE	System - skel. dir.: AE	Input	Error messages
PACGGN	Admin Database - Dbase dir.: AN	Input	Administration Database index
PACGGR	Admin Database - Dbase dir.: AR	Input	Administration Database data
PACGGU	Admin Database - Dbase dir.: GU	Input	Administration Database users
PAC7AN	Dbase dir.: AN	Input	Development Database index
PAC7AR	Dbase dir.: AR	Input	Development Database data
PAC7MB	User input	Input	User transactions
PAC7BM	Tmp dir.: WBM	Output	User assignment
PAC7EX	Tmp dir.: WEX	Output	Extracted transactions
PAC7ET	User dir.: PUPMET450	Report	Environment types validation
PAC7DD	User dir.: PUPMDD450	Report	Authorizations control

Generation of purge transactions: PCM460

This step checks the extracted transactions, deletes the duplicated transactions and generates purge transactions.

Code	Physical name	Type	Label
PAC7AE	System - Skel. dir.: AE	Input	Error messages
PAC7AN	Database dir.: AN	Input	Development Database index
PAC7AR	Database dir.: AR	Input	Development Database data
PAC7BM	Tmp dir.: WBM	Input	User assignment
PAC7EX	Tmp dir.: WEX	Input	Extracted transactions
PAC7GY	User dir.: PUPMMY	Output	generated purge transactions

Transactions formatting: PAF900

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

Update of the Development Database: PACA15

Code	Physical name	Type	Label
PAC7AR	Database dir.: AR	Output	Development Database Data file
PAC7AN	Database dir.: AN	Output	Development Database index
PAC7AY	Database dir.: AY	Output	Development Database extension
PAC7AJ	Journal dir.: AJ	Output	Development Database journal
PAC7AE	System - Skel. dir.: AE	Input	Error messages
PACGGN	Admin. DBase - Base dir.: AN	Input	Administration Database Index file
PACGGR	Admin. DBase - Base dir.: AR	Input	Administration Database Data file
PACGGY	Admin. DBase - Base dir.: AY	Input	Administration Database Extension

Code	Physical name	Type	Label
PACGGU	Admin. DBase - Base dir.: GU	Input	Administration Database users
PAC7DC	NUL	Input	DSMS file of Development Database Elements
PAC7ME	Tmp dir.: WME	Input	Work file
PAC7MV	Tmp dir.: WMV	Input	Update transactions
PAC7RB	NUL	Output	UPDT erroneous transactions (length=80)
PAC7RY	NUL	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, no error
- 2: Warning
- 4: Error

PUPM - Execution JCL

```
' -----------------------------------------------------
'      VISUALAGE PACBASE
'
' -----------------------------------------------------
' PURGE
' -----------------------------------------------------
'
' INPUT     : USER IDENTIFICATION
' COL 2     : "*"
' COL 3     : USER CODE
' COL 11    : PASSWORD
' -----------------------------------------------------
<job id=PUPM>

<script language="VBScript">
Dim MyProc
MyProc = "PUPM"
</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" , "PCM450"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
Call BvpEnv("PCM450","PAC7DD",Rep_USR & "\PUPMDD450.txt")
Call BvpEnv("PCM450","PAC7ET",Rep_USR & "\PUPMET450.txt")
WshEnv("PAC7MB") = Fic_Input
Call BvpEnv("PCM450","PAC7EX",Rep_TMP & "\WEX.tmp")
Call BvpEnv("PCM450","PAC7BM",Rep_TMP & "\WBM.tmp")
Call RunCmdLog ("BVPCM450")
Call Err_Cod(Return , 0 , "PCM450")


Call Msg_Log (Array("1022" , "PCM460"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
Call BvpEnv("PCM460","PAC7GY",Rep_USR & "\PUPMMY.txt")
Call BvpEnv("PCM460","PAC7EX",Rep_TMP & "\WEX.tmp")
Call BvpEnv("PCM460","PAC7BM",Rep_TMP & "\WBM.tmp")
Call RunCmdLog ("BVPCM460")
Call Err_Cod(Return , 0 , "PCM460")

Call Msg_Log (Array("1022" , "PAF900"))
'-----------------------------------
Call BvpEnv("PAF900","PAC7GY",Rep_USR & "\PUPMMY.txt")
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
Call BvpEnv("PAF900","PAC7ME",Rep_TMP & "\WME.tmp")
Call BvpEnv("PAF900","PAC7MW",Rep_TMP & "\WMW.tmp")
Call BvpEnv("PAF900","PAC7MV",Rep_TMP & "\WMV.tmp")
Call BvpEnv("PAF900","PAC7MX",Rep_TMP & "\WMX.tmp")
Call BvpEnv("PAF900","PAC7MY",Rep_TMP & "\WMY.tmp")
Call RunCmdLog ("BVPAF900")
Call Err_Cod(Return , 0 , "PAF900")

Call Msg_Log (Array("1022" , "PACA15"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AJ") = Rep_JOURNAL & "\AJ"
WshEnv("PAC7AN") = Rep_BASE & "\AN"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
WshEnv("PAC7AY") = Rep_BASE & "\AY"
```

```
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGY") = Rep_ABASE & "\AY"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
WshEnv("PAC7DC") = Rep_TMP & "\WDC.tmp"
'PAC7DC not used, on default
Call BvpEnv("PACA15","PAC7IE",Rep_USR & "\PUPMIEA15.txt")
Call BvpEnv("PACA15","PAC7IF",Rep_USR & "\PUPMIFA15.txt")
WshEnv("SEMLOCK") = Rep_BASE & "\LO"
WshEnv("SEMADMIN") = Rep_ABASE & "\LO"
Call BvpEnv("PACA15","PAC7ME",Rep_TMP & "\WME.tmp")
Call BvpEnv("PACA15","PAC7MV",Rep_TMP & "\WMV.tmp")
WshEnv("PAC7RB") = Rep_TMP & "\WRB.tmp"
'PAC7RB not used, on default
WshEnv("PAC7RY") = Rep_TMP & "\WRY.tmp"
'PAC7RY not used, on default

Call RunCmdLog ("BVPACA15")
If Return = 2 Then
Call Msg_Log (Array("1061"))
End If
If Return = 4 Then
Call Msg_Log (Array("1060"))
End If
Call Err_Cod(Return , 4 , "PACA15")


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


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


</script>
</job>
```

Pac/Transfer

Introduction

The purpose of the Pac/Transfer facility is to provide an easy versioning of the developments made in the Development Database; it automates the transfers of update transactions between two sessions or more.

Pac/Transfer scans the Development Database archived Journal file and reads the Administration Database where the processing parameters are stored.

These parameters define one or more source environments. Each source environment can correspond to one or more target environments.

Pac/Transfer selects, from the archived Journal file, the transactions that match the criteria defined via these parameters.

Pac/Transfer then generates update transactions for the target environment(s) defined in these parameters.

These transactions are used by the VA Pac batch update procedure (UPDT). If the Development 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 into another validation-dedicated session, and, if necessary, into 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).

Note

The transfer parameters are stored in the Administration Database, for all the Development Databases managed by the Administrator. The list of these Databases is itself defined in the Administration Database.

The Development Database notion is then 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, as indicated above.

Chronology of Processing

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.

Update of Transfer Parameters

TRUP - Introduction

Pac/Transfer processing is based on user-defined parameters stored in the Administration Database. These parameters are used in all the Pac/Transfer procedures.

These parameters must be created -- via a TRUP execution -- prior to any Pac/transfer operation. Any change to one of these parameters must be followed by a new TRUP execution.

Several sets of transfer parameters, called Transfer Sets, may be defined.

A Transfer Set is linked to a Development Database.

A Transfer Set may be used for several Development Databases.

When executing the TRUP procedure, an access to the Development Database is performed in order to validate the parameters which make up the transfer set, and then these parameters are stored in the Administration Database.

The transfer set identifier stored in this Database is constituted of the 'Development Database logical code' + 'transfer set code'. The Development Database logical code is automatically assigned to the identifier during the TRUP procedure processing.

The other Pac/Transfer batch procedures follow the same principle: they search for the processing parameters in the Administration Database with an identifier constituted of the Development Database logical code and the transfer set code.

By defining several Transfer Sets, you obtain flexible transfer operations, fully adaptable to your own requirements.

The transfer parameters, described below, define one Transfer 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 related to the 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 the Libraries of the Development Database are taken into account for the requested source session; these Libraries are also the targets of the transfers.

  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 among the selected target Libraries, specify its code explicitly.

If you specify several target Libraries, the transactions related 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 the transferred transactions will be SIX times larger than the volume of the selected transactions.

- User

  As a default, the transactions entered by ANY Development 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 Development Databases under DSMS control only.

  By default, the 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.

  All the transactions can be transferred under a single target user code.

  Note: This option overrides any target user selection described above.

Execution conditions

None.

Printed report

Printout of the content of the parameter file.

TRUP - User Input

User identification line (required)

Position	Length	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.

Position	Length	Value	Meaning
1	1		Action code
		'C'	Creation
		'M'	Modification
		'D'	Deletion
		' '	Creation or modification depending on the Database status
		'X'	Equal to ' ', with no 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

A Transfer Set must include at least one selection line of this type.

Position	Length	Value	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	'GS'	Line type
4	4		Source session (required)
13	2		Line number (two lines only are authorized)

Position	Length	Value	Meaning
		'00'	First line for the 9 first target sessions (default value)
		'01'	Continuation line for the 9 following target sessions, if necessary (the target session number is limited to 18: the input in position 1 to 7 on the first line must be repeated on continuation line).
15	3		Sequence number of transfers (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)

Library selection line

Position	Length	Value	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

Position	Length	Value	Meaning
1	1		Action code
		'C'	Creation
		'M'	Modification
		'D'	Deletion

Position	Length	Value	Meaning
		' '	Creation or modification depending on 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

Position	Length	Value	Meaning
1	1		Action code:
		'C'	Creation
		'M'	Modification
		'D'	Deletion
		' '	Creation or modification depending on 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:

- on each selection type for a Transfer Set,
- on a whole Transfer Set,

- on all the Sets of a Development Database.

Position	Length	Value	Meaning
1	1	'B'	Multiple deletion request
2	2	'GA'	Deletion of the whole Set
		'GS'	Deletion of Set lines: 'GS' 'GB' GC' and 'GU'
		'GB'	Deletion of 'GB' Set lines
		'GU'	Deletion of 'GU' Set lines
		'GC'	Deletion of 'GC' Set lines
4	10		Only if value 'GA' in columns 2 and 3.
		llllllllll	Set code
		'**********'	Deletion of all the Sets of a Development Database

TRUP - Description of Steps

Update of the Administration Database: PTUG20

This step updates the Administration Database in order to store the selection parameters.

Code	Physical name	Type	Label
PAC7AR	Database dir. : AR	Input	Development Database data file
PAC7AN	Database dir. : AN	Input	Development Database index file
PAC7AY	NUL	Output	Development Database extension data
PACGGU	Admin Database - Base dir. : GU	Input	Users file
PAC7AE	System - Skel dir. : AE	Input	Error messages file
PACGGR	Admin Database - Base dir. : AR	Input/ Output	Administration Database data file
PACGGN	Admin Database - Base dir. : AN	Input/ Output	Administration Database index file
PACGGY	Admin Database - Base dir. : AY	Input/ Output	Administration Database extension data file
PACGGJ	Journal dir. : AJ	Input/ Output	Administration Database Journal file

Code	Physical name	Type	Label
PAC7MC	User input	Input	Parameters update transactions file
PAC7ME	Tmp dir. : WME	Input/ Output	Work file
PAC7MY	Tmp dir. : WMY	Input/ Output	Work file
PAC7TB	Tmp dir. : WTB	Input/ Output	Work file
PAC7BM	Tmp dir. : WBM	Output	Parameters print request file
PAC7ET	User dir. : TRUPETG20	Report	Input validation
PAC7IE	User dir. : TRUPIEG20	Report	Administration Database update report
PAC7IF	User dir. : TRUPIFG20	Report	Administration Database update errors
PAC7DD	User dir. : TRUPDDG20	Report	Output report: validation of '*' line in relation to the Development Database
PAC7DE	User dir. : TRUPDEG20	Report	Output report: validation of '*' line in relation to the Administration Database

Extraction of the Administration Database: PTUG30

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

Code	Physical name	Type	Label
PAC7MB	Tmp dir. : WMB	Input	Extraction request file
PAC7GL	Tmp dir. : WGL	Output	Target sessions list
PAC7UY	Tmp dir. : WUY	Output	Reduced parameters file
PAC7DD	User dir. : TRUPDDG30	Report	'*' line validation report
PAC7TK	Tmp dir. : WTK	Input/Output	Work file

Printing of selection parameters: PTUG31

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages file
PAC7UY	Tmp dir. : WUY	Input	Reduced parameters file
PAC7MB	Tmp dir. : WMB	Input	Extraction request file
PAC7ET	User dir. : TRUPETG31	Report	Printing of the selection parameters list

Printing of the target Sessions file: PTUG32

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7MB	Tmp dir : WMB	Input	Extraction request file
PAC7GL	Tmp dir. : WGL	Input	Target session
PAC7ET	User dir. : TRUPETG32	Report	Printing of the target sessions list

TRUP - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'        PAC/TRANSFER:
'        UPDATE OF THE TRANSFER PARAMETERS
' ------------------------------------------------------
'
' PAC/TRANSFER - PROCESSING IS BASED ON THE USER-DEFINED
' PARAMETERS STORED IN THE ADMINISTRATION DATABASE.
' THESE PARAMETERS CONTROL THE VARIOUS PROCESSES OF THE
' PROCEDURES OF FACILITY.
'
```

```
' ------------------------------------------------------
'
<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") = Rep_TMP & "\NUL.tmp"
'PAC7AY not used, on default
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGJ") = Rep_AJOURNAL & "\AJ"
WshEnv("PACGGY") = Rep_ABASE & "\AY"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
Call BvpEnv("PTUG20","PAC7BM",Rep_TMP & "\WMB.tmp")
Call BvpEnv("PTUG20","PAC7DD",Rep_USR & "\TRUPDDG20.txt")
Call BvpEnv("PTUG20","PAC7DE",Rep_USR & "\TRUPDEG20.txt")
Call BvpEnv("PTUG20","PAC7ET",Rep_USR & "\TRUPETG20.txt")
Call BvpEnv("PTUG20","PAC7IE",Rep_USR & "\TRUPIEG20.txt")
Call BvpEnv("PTUG20","PAC7IF",Rep_USR & "\TRUPIFG20.txt")
WshEnv("PAC7MC") = Fic_Input
Call BvpEnv("PTUG20","PAC7ME",Rep_TMP & "\WME.tmp")
Call BvpEnv("PTUG20","PAC7MY",Rep_TMP & "\WMY.tmp")
Call BvpEnv("PTUG20","PAC7TB",Rep_TMP & "\WTB.tmp")
Call RunCmdLog ("BVPTUG20")
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") = Rep_TMP & "\NUL.tmp"
'PAC7AY not used, on default
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
Call BvpEnv("PTUG30","PAC7DD",Rep_USR & "\TRUPDDG30.txt")
Call BvpEnv("PTUG30","PAC7GL",Rep_TMP & "\WGL.tmp")
Call BvpEnv("PTUG30","PAC7MB",Rep_TMP & "\WMB.tmp")
```

```
Call BvpEnv("PTUG30","PAC7TK",Rep_TMP & "\WTK.tmp")
Call BvpEnv("PTUG30","PAC7UY",Rep_TMP & "\WUY.tmp")
Call RunCmdLog ("BVPTUG30")
Call Err_Cod(Return , 0 , "PTUG30")

Call Msg_Log (Array("1022" , "PTUG31"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AR") = Rep_ABASE & "\AR"
Call BvpEnv("PTUG31","PAC7GL",Rep_TMP & "\WGL.tmp")
Call BvpEnv("PTUG31","PAC7UY",Rep_TMP & "\WUY.tmp")
Call BvpEnv("PTUG31","PAC7ET",Rep_USR & "\TRUPETG31.txt")
Call RunCmdLog ("BVPTUG31")
Call Err_Cod(Return , 0 , "PTUG31")


Call Msg_Log (Array("1022" , "PTUG32"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AR") = Rep_ABASE & "\AR"
Call BvpEnv("PTUG32","PAC7GL",Rep_TMP & "\WGL.tmp")
Call BvpEnv("PTUG32","PAC7ET",Rep_USR & "\TRUPETG32.txt")
Call RunCmdLog ("BVPTUG32")
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 prints all the parameters of each Development Database and transfer set.

It is possible to print all the parameters, or to print the parameters of one Development Database only.

TRED - User Input

User identification line (required).

Position	Length	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

Validation of the processing request: PTUG28

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database data
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database index
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database users
PAC7AR	Admin Database - Base dir. : AR	Input	Administration Database data
PAC7MB	User input	Input	Parameters print request
PAC7BM	Tmp dir. : WMB	Output	Parameters print request
PAC7DD	User dir. : TREDDDG28	Report	Output report: user code validity check
PAC7ET	User dir. : TREDETG28	Report	Output report: validation of print request

Extraction of the Administration Database: PTUG30

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

Code	Physical name	Type	Label
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database index
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database data
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database users
PAC7AN	Admin Database - Base dir. : AN	Input	Administration Database index
PAC7AR	Admin Database - Base dir. : AR	Input	Administration Database data
PAC7AY	NUL	Input	Administration Database extension data
PAC7MB	Tmp dir. : WMB	Input	Print requests file
PAC7GL	Tmp dir. : WGL	Output	Target sessions file
PAC7UY	Tmp dir. : WUY	Output	Reduced parameters file
PAC7TK	Tmp dir. : WTK	Input/Output	Work file
PAC7DD	User dir. : TREDDDG30	Report	'*' line validation report

Printing of selection parameters: PTUG31

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7UY	Tmp dir. : WUY	Input	Reduced parameters
PAC7MB	Tmp dir : WMB	Input	Extraction requests file
PAC7ET	User dir. : TREDETG31	Report	List of selection parameters

Printing of target Sessions list: PTUG32

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7GL	Tmp dir. : WGL	Input	Target sessions
PAC7MB	Tmp dir. : WMB	Input	Extraction request file
PAC7ET	User dir. : TREDETG32	Report	List of target sessions

TRED - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'        PAC/TRANSFER -
'        EDITING THE DATABASE PARAMETERS
' ------------------------------------------------------
'
' FOR ALL THE DATABASE OR ONE DATABASE
'
' INPUT :
'  -  USER IDENTIFICATION LINE (REQUIRED)
'
' ------------------------------------------------------
'
<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"
Call BvpEnv("PTUG28","PAC7BM",Rep_TMP & "\WMB.tmp")
Call BvpEnv("PTUG28","PAC7DD",Rep_USR & "\TREDDDG28.txt")
Call BvpEnv("PTUG28","PAC7ET",Rep_USR & "\TREDETG28.txt")
WshEnv("PAC7MB") = Fic_Input
Call RunCmdLog ("BVPTUG28")
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") = Rep_TMP & "\NUL.tmp"
'PAC7AY not used, on default
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
```

```
Call BvpEnv("PTUG30","PAC7DD",Rep_USR & "\TREDDDG30.txt")
Call BvpEnv("PTUG30","PAC7GL",Rep_TMP & "\WGL.tmp")
Call BvpEnv("PTUG30","PAC7MB",Rep_TMP & "\WMB.tmp")
Call BvpEnv("PTUG30","PAC7TK",Rep_TMP & "\WTK.tmp")
Call BvpEnv("PTUG30","PAC7UY",Rep_TMP & "\WUY.tmp")
Call RunCmdLog ("BVPTUG30")
Call Err_Cod(Return , 0 , "PTUG30")

Call Msg_Log (Array("1022" , "PTUG31"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AR") = Rep_ABASE & "\AR"
Call BvpEnv("PTUG31","PAC7UY",Rep_TMP & "\WUY.tmp")
Call BvpEnv("PTUG31","PAC7ET",Rep_USR & "\TREDETG31.txt")
Call RunCmdLog ("BVPTUG31")
Call Err_Cod(Return , 0 , "PTUG31")


Call Msg_Log (Array("1022" , "PTUG32"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AR") = Rep_ABASE & "\AR"
Call BvpEnv("PTUG32","PAC7GL",Rep_TMP & "\WGL.tmp")
Call BvpEnv("PTUG32","PAC7ET",Rep_USR & "\TREDETG32.txt")
Call RunCmdLog ("BVPTUG32")
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>
```

Compression of Archived Journal

TRJC - Introduction

From the archived Journal of the Development Database, the TRJC procedure (which is optional) produces a compressed Journal which contains useful transactions only; the intermediate transactions, useless for the transfer, are purged.

To ensure data integrity, the transactions related to certain entities cannot be compressed ; this applies to the lines whose codes are: G, H, L4, V3, Y1 to Y6, and 4

User input may include an interval of dates and/or session numbers in order to limit transfer processing to the archived Journal transactions which belong 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 with 'useful' transactions only.

Printed report

Statistical data on the TRJC execution.

TRJC - User Input

User identification line (required).

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

Options

Position	Length	Value	Meaning
1	1		Deletion of user codes
		'0'	Yes

Position	Length	Value	Meaning
		'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 phase): PTUG04

Code	Physical Name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7AR	Database dir. : AR	Input	Development Database data
PAC7AN	Database dir. : AN	Input	Development Database index
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database index
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database data
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database uses
PAC7MB	User input	Input	User input
PAC7BM	Tmp dir. : WMB	Output	User input
PAC7DD	User dir. : TRJCDDG04	Report	Batch procedures' execution report

Compression (first phase): PTUG05

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7PJ	Save dir. : PJ	Input	Sequential image of the Development Database
PAC7AN	Base dir. : AN	Input	Development Database index

Code	Physical name	Type	Label
PACGGN	Admin Base - Base dir. : AN	Input	Administration Database index
PACGGR	Admin Base - Base dir. : AR	Input	Administration Database data
PACGGU	Admin Base - Base dir. : GU	Input	Administration Database users
PAC7MB	Tmp dir. : WMB	Input	User Input
PAC7GP	Tmp dir. : WGP	Output	Temporary journal
PAC7ET	User dir. : TRJCETG05	Report	Input validation
PAC7DD	User dir. : TRJCDDG05	Report	Batch procedures error status

Compression (second phase): PTUG06

Code	Physical name	Type	Label
PAC7GP	Tmp dir. : WGP	Input	Temporary Journal
PAC7PK	Tmp dir. : WPK	Output	Compressed sequential journal

Classification of deletions/creations: PTUG07

Code	Physical name	Type	Label
PAC7AN	Database dir. : AN	Input	Development Database index
PAC7PK	Tmp dir. : WPK	Input	Temporary journal
PAC7PL	Save dir. : JT	Output	Compressed sequential journal

TRJC - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'       PAC/TRANSFER -
'       COMPRESSION OF ARCHIVED JOURNAL
' ------------------------------------------------------
'
'
' FROM THE DATABASE 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.
```

```
'
' INPUT :
'  -  USER IDENTIFICATION LINE (REQUIRED)
'  -  COMMAND LINE :
'  COL 1  :  DELETION OF USER CODES:
'            "0"   YES
'            "1"   NO
'  COL 2  :  DELETION OF CHANGE NUMBERS:
'            "0"   YES
'            "1"   NO
'  COL 3  :  (4 CAR.) START SESSION NUMBER
'  COL 7  :  (4 CAR.) END SESSION NUMBER
'
'  COL 11 :  (8 CAR.) START DATE IN THE FORM CCYYMMDD
'  COL 19 :  (8 CAR.) END DATE IN THE FORM CCYYMMDD
' -----------------------------------------------------
'
<job id=TRJC>

<script language="VBScript">
Dim MyProc
MyProc = "TRJC"
</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" , "PTUG04"))
'-----------------------------------
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("PAC7MB") = Fic_Input
Call BvpEnv("PTUG04","PAC7DD",Rep_USR & "\TRJCDDG04.txt")
Call BvpEnv("PTUG04","PAC7BM",Rep_TMP & "\WMB.tmp")
Call RunCmdLog ("BVPTUG04")
Call Err_Cod(Return , 0 , "PTUG04")

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"
Call BvpEnv("PTUG05","PAC7DD",Rep_USR & "\TRJCDDG05.txt")
Call BvpEnv("PTUG05","PAC7ET",Rep_USR & "\TRJCETG05.txt")
Call BvpEnv("PTUG05","PAC7GP",Rep_TMP & "\WGP.tmp")
```

```
Call BvpEnv("PTUG05","PAC7MB",Rep_TMP & "\WMB.tmp")
WshEnv("PAC7PJ") = Rep_SAVE & "\PJ"
Call RunCmdLog ("BVPTUG05")
Call Err_Cod(Return , 0 , "PTUG05")

Call Msg_Log (Array("1022" , "PTUG06"))
'-----------------------------------
Call BvpEnv("PTUG06","PAC7GP",Rep_TMP & "\WGP.tmp")
Call BvpEnv("PTUG06","PAC7PK",Rep_TMP & "\WPK.tmp")
Call RunCmdLog ("BVPTUG06")
Call Err_Cod(Return , 0 , "PTUG06")

Call Msg_Log (Array("1022" , "PTUG07"))
'-----------------------------------
WshEnv("PAC7AN") = Rep_BASE & "\AN"
Call BvpEnv("PTUG07","PAC7PK",Rep_TMP & "\WPK.tmp")
WshEnv("PAC7PL") = Rep_SAVE & "\JT"
Call RunCmdLog ("BVPTUG07")
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>
```

Creation of the Transfer File

TRPF - Introduction

From the archived Journal --whether compressed or not, depending on the site and according to the contents of the Parameters file-- the TRPF procedure produces a Transfer file, which has the following characteristics:

- Only the transactions which meet the source selection parameters (sessions, Libraries, users, Changes) are processed.
- The values of the selected parameters are replaced with those of the target parameters specified in the processed Set.
- 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 related to a Development Database.

Execution conditions

None.

Result

The TRPF procedure produces a Transfer file and a parameters file reduced and adapted to the processing request (UY).

These two files will be used by the TRRP procedure.

TRPF - User Input

User identification line (required)

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

Selection of the Transfer Set to be processed (required)

Position	Length	Value	Meaning
2	2	'LT'	
4	10	llllllllll	Code of Transfer Set to process
		'**********'	Selection of all Sets

Note

The selection of all the Sets necessarily implies that only one LT-type line be entered (with the value '**********' in Positions 4 to 13).

TRPF - Description of Steps

Validation of the processing request: PTUG27

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

Code	Physical name	Type	Label
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database users
PAC7AN	Database dir. : AN	Input	Development Database index
PAC7AR	Database dir. : AR	Input	Development Database data
PAC7MB	User input	Input	Parameters extraction request
PAC7BM	Tmp dir. : WBM	Output	Parameters extraction request
PAC7DD	User dir. : TRPFDDG27	Report	User code validation report check
PAC7ET	User dir. : TRPFETG27	Report	Extraction request validation report

Extraction of the Administration Database: PTUG30

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database index
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database data
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database users
PAC7AN	Admin Database - Base dir. : AN	Input	Administration Database index
PAC7AR	Admin Database - Base dir. : AR	Input	Administration Database data
PAC7AY	NUL	Input	Administration Database extension data
PAC7MB	Tmp dir. : WBM	Input	Extraction request
PAC7GL	Tmp dir. : WGL	Output	List of target sessions
PAC7UY	Tmp dir. : WUY	Output	Reduced parameters for TRRP processing
PAC7DD	User dir. : TRPFDDG30	Report	'*' line validation report
PAC7TK	Tmp dir. : WTK	Output	Work file

Indexation of UY file: PTUG39

Code	Physical name	Type	Label
PAC7YU	Tmp dir. : WUY	Input	Limited parameters for TRRP processes
PAC7UY	User dir. : UY	Output	Limited parameters

Creation of the transfers file: PTUG50

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database index
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database data
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database users
PAC7AR	Database dir. : AR	Input	Development Database data
PAC7JT	Save dir. : PJ	Input	Sequential or compressed Journal
PAC7MB	Tmp dir. : WBM	Input	User input
PAC7UY	User dir. : UY	Input	Reduced parameters
PAC7TJ	Save dir. : TJ	Output	Transfers file
PAC7DD	User dir. : TRPFDDG50	Report	User line validation report
PAC7ER	User dir. : TRPFERG50	Report	List of user input
PAC7ET	User dir. : TRPFETG50	Report	Statistics on transfers

TRPF - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'        PAC/TRANSFER -
'        CREATING THE TRANSFER FILE
' ------------------------------------------------------
'
' FROM THE ARCHIVED JOURNAL THE TRPF PROCEDURE PRODUCES
' A TRANSFER FILE.
'
' INPUT :
'  -  USER IDENTIFICATION LINE (REQUIRED)
'  -  COMMAND LINE :
'  COL 2  : "LT"
'  COL 4  : (5 CAR.) TRANSACTION SET FOR PROCESSING CODE.
```

```
'           IF SELECTION OF ALL SETS "*****"
'
' ------------------------------------------------------
'
<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"
Call BvpEnv("PTUG27","PAC7BM",Rep_TMP & "\WBM.tmp")
Call BvpEnv("PTUG27","PAC7DD",Rep_USR & "\TRPFDDG27.txt")
Call BvpEnv("PTUG27","PAC7ET",Rep_USR & "\TRPFETG27.txt")
WshEnv("PAC7MB") = Fic_Input
Call RunCmdLog ("BVPTUG27")
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") = Rep_TMP & "\NUL.tmp"
'PAC7AY not used, on default
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
Call BvpEnv("PTUG30","PAC7DD",Rep_USR & "\TRPFDDG30.txt")
Call BvpEnv("PTUG30","PAC7GL",Rep_TMP & "\WGL.tmp")
Call BvpEnv("PTUG30","PAC7MB",Rep_TMP & "\WBM.tmp")
Call BvpEnv("PTUG30","PAC7TK",Rep_TMP & "\WTK.tmp")
Call BvpEnv("PTUG30","PAC7UY",Rep_TMP & "\WUY.tmp")
Call RunCmdLog ("BVPTUG30")
Call Err_Cod(Return , 0 , "PTUG30")

Call Msg_Log (Array("1022" , "PTUG39"))
'-----------------------------------
Call BvpEnv("PTUG39","PAC7YU",Rep_TMP & "\WUY.tmp")
Call BvpEnv("PTUG39","PAC7UY",RepT_USR & "\UY")
Call RunCmdLog ("BVPTUG39")
```

```
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"
Call BvpEnv("PTUG50","PAC7DD",Rep_USR & "\TRPFDDG50.txt")
Call BvpEnv("PTUG50","PAC7ER",Rep_USR & "\TRPFERG50.txt")
Call BvpEnv("PTUG50","PAC7ET",Rep_USR & "\TRPFETG50.txt")
WshEnv("PAC7JT") = Rep_SAVE & "\PJ"
Call BvpEnv("PTUG50","PAC7MB",Rep_TMP & "\WBM.tmp")
WshEnv("PAC7TJ") = Rep_SAVE & "\TJ"
Call BvpEnv("PTUG50","PAC7UY",RepT_USR & "\UY")
Call RunCmdLog ("BVPTUG50")
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 TRUD procedure must be used when the Development Database is under DSMS control, and when source criteria include a selected Change number.

TRDU can be applied to one selected Set or all the Sets related to a Development Database.

The VisualAge Pacbase authorizations indicated for the target Change(s) must include the authorizations for 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 recognizes 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 on 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 outputs two files:

- A DSMS update transactions file to allow target Change(s) to accept updates made on the source Change(s).

  Also, all the 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 back to their initial state.

  This update must be executed AFTER the transfers are introduced in the Development Database.

Execution conditions

None.

Results

Two DSMS batch update transactions files, one of which should be applied before the transfers, the other after all the transfers.

TRDU - User Input

User identification line (required).

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

Selection of the transfer set (required)

Position	Length	Value	Meaning
2	2	'LT'	
4	10	llllllllll	Selected Transaction Set code
		'**********'	Selection of all Sets

Only one LT-type line is required.

TRDU - Description of Steps

Validation of processing request: PTUG26

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database data
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database index
PAC7GU	Admin Database - Base dir. : GU	Input	Administration Database index
PAC7AN	Database dir. : AN	Input	Development Database index
PAC7AR	Database dir. : AR	Input	Development Database data
PAC7MB	User input	Input	Parameters extraction request
PAC7BM	Tmp dir. : WBM	Output	Parameters extraction request
PAC7ET	User dir. : TRDUETG26	Report	User code validation report code
PAC7DD	User dir. : TRDUDDG26	Report	Print request validation report

Extraction of the Administration Database: PTUG30

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

Code	Physical name	Type	Label
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database users
PAC7AN	Admin Database - Base dir. : AN	Input	Administration Database index
PAC7AR	Admin Database - Base dir. : AR	Input	Administration Database data
PAC7AY	NUL	Input	Administration Database extension data
PAC7MB	Tmp dir. : WBM	Input	Extraction request
PAC7GL	Tmp dir. : WGL	Output	Target sessions
PAC7UY	Tmp dir. : WUY	Output	Reduced parameters
PAC7DD	User dir. : TRDUDDG30	Report	Report output: '*' line validation
PAC7TK	Tmp dir. : WTK	Input/Output	Work file

Indexation of UY file: PTUG39

Code	Physical name	Type	Label
PAC7YU	Tmp dir. : WUY	Input	Limited parameters for TRRP processes
PAC7UY	User dir. : UY	Output	Limited parameters

Selection of sets: PTUG42

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database index
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database data
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database users
PAC7AR	Database dir. : AR	Input	Development Database data
PAC7UY	User dir. : UY	Input	Reduced parameters
PAC7MB	Tmp dir. : WBM	Input	User input
PAC7BM	Tmp dir. : WBM2	Output	Sets file
PAC7DD	User dir. : TRDUDDG42	Report	User code validation report

Code	Physical name	Type	Label
PAC7ET	User dir. : TRDUETG42	Report	Extraction validation report

DSMS preparation before the transfers: PTUG44

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7AR	Database dir. : AR	Input	Development Database data
PAC7UY	User dir. : UY	Input	Limited parameters
PACDDC	Database dir. : DC	Input	Elements of the product (DSMS)
PAC7MB	Tmp dir. : WBM2	Input	Batch transactions
PAC7CI	Tmp dir. : WCI	Output	Delete transactions of target after the transfer
PAC7SI	Tmp dir. : WSI	Output	Delete transactions of the source/target initial state
PAC7GC	Tmp dir. : WGC	Output	Preparation of the authorizations of the target Changes
PAC7ET	User dir. : TRDUETG44	Report	Report

Generation of target change transactions: PTUG46

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7AR	Base dir. : AR	Input	Development Database data
PAC7GC	Tmp dir. : WGC	Input	Preparation of target change authorizations
PAC7CC	Tmp dir. : WCC	Output	Target creation transactions before transfer
PAC7SC	Tmp dir. : WSC	Output	Target deletion transactions after transfer
PAC7ET	User dir. : TRDUETG46	Report	Report output

Copy/Merge SI + CC ==> TRDUAV :

Code	Physical name	Type	Label
SI	Tmp dir. : WSI	Input	Deletion transactions of the source/target initial status
CC	Tmp dir. : WCC	Input	Target creation transactions before transfer
TRDUAV	User dir. : TRDUAV	Output	Transactions for DUPT before the Development Database update

Copy/merge SC + CI ==> TRDUAP :

Code	Physical name	Type	Label
SC	Tmp dir. : WSC	Input	Target deletion transactions after transfer
CI	Tmp dir. : WCI	Input	Creation transactions after transfer
TRDUAP	User dir. : TRDUAP	Output	transactions for DUPT after Development Database update

TRDU - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'       PAC/TRANSFER -
'       PREPARING THE DSMS ENVIRONMENT
' ------------------------------------------------------
'
' THE DSMS-ENVIRONMENT PREPARATION PROCEDURE
' (TRDU) MUST BE USED WHEN THE DEVELOPMENT DATABASE
' IS UNDER DSMS CONTROL, AND WHEN SOURCE CRITERIA INCLUDE
' A SELECTED CHANGE NUMBER.
'
' INPUT :
'  -  USER IDENTIFICATION LINE (REQUIRED)
'  -  COMMAND LINE :
'  COL 2  : "LT"
'  COL 4  : (10 CAR.) SELECTED TRANSACTION SET CODE.
'           IF SELECTION OF ALL SETS "**********"
'
' ------------------------------------------------------
'
<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"
Call BvpEnv("PTUG26","PAC7BM",Rep_TMP & "\WBM.tmp")
Call BvpEnv("PTUG26","PAC7DD",Rep_USR & "\TRDUDDG26.txt")
Call BvpEnv("PTUG26","PAC7ET",Rep_USR & "\TRDUETG26.txt")
WshEnv("PAC7MB") = Fic_Input
Call RunCmdLog ("BVPTUG26")
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") = Rep_TMP & "\NUL.tmp"
'PAC7AY not used, on default
WshEnv("PACGGN") = Rep_ABASE & "\AN"
WshEnv("PACGGR") = Rep_ABASE & "\AR"
WshEnv("PACGGU") = Rep_ABASE & "\GU"
Call BvpEnv("PTUG30","PAC7DD",Rep_USR & "\TRDUDDG30.txt")
Call BvpEnv("PTUG30","PAC7GL",Rep_TMP & "\WGL.tmp")
Call BvpEnv("PTUG30","PAC7MB",Rep_TMP & "\WBM.tmp")
Call BvpEnv("PTUG30","PAC7TK",Rep_TMP & "\WTK.tmp")
Call BvpEnv("PTUG30","PAC7UY",Rep_TMP & "\WUY.tmp")
Call RunCmdLog ("BVPTUG30")
Call Err_Cod(Return , 0 , "PTUG30")


Call Msg_Log (Array("1022" , "PTUG39"))
'-----------------------------------
Call BvpEnv("PTUG39","PAC7YU",Rep_TMP & "\WUY.tmp")
Call BvpEnv("PTUG39","PAC7UY",RepT_USR & "\UY")
Call RunCmdLog ("BVPTUG39")
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"
Call BvpEnv("PTUG42","PAC7DD",Rep_USR & "\TRDUDDG42.txt")
Call BvpEnv("PTUG42","PAC7ET",Rep_USR & "\TRDUETG42.txt")
Call BvpEnv("PTUG42","PAC7BM",Rep_TMP & "\WBM2.tmp")
Call BvpEnv("PTUG42","PAC7MB",Rep_TMP & "\WBM.tmp")
Call BvpEnv("PTUG42","PAC7UY",RepT_USR & "\UY")
Call RunCmdLog ("BVPTUG42")
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"
Call BvpEnv("PTUG44","PAC7MB",Rep_TMP & "\WBM2.tmp")
Call BvpEnv("PTUG44","PAC7CI",Rep_TMP & "\WCI.tmp")
Call BvpEnv("PTUG44","PAC7GC",Rep_TMP & "\WGC.tmp")
Call BvpEnv("PTUG44","PAC7SI",Rep_TMP & "\WSI.tmp")
Call BvpEnv("PTUG44","PAC7UY",RepT_USR & "\UY")
Call BvpEnv("PTUG44","PAC7ET",Rep_USR & "\TRDUETG44.txt")
Call RunCmdLog ("BVPTUG44")
Call Err_Cod(Return , 0 , "PTUG44")


Call Msg_Log (Array("1022" , "PTUG46"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
Call BvpEnv("PTUG46","PAC7ET",Rep_USR & "\TRDUETG46.txt")
Call BvpEnv("PTUG46","PAC7CC",Rep_TMP & "\WCC.tmp")
Call BvpEnv("PTUG46","PAC7GC",Rep_TMP & "\WGC.tmp")
Call BvpEnv("PTUG46","PAC7SC",Rep_TMP & "\WSC.tmp")
Call RunCmdLog ("BVPTUG46")
Call Err_Cod(Return , 0 , "PTUG46")

Call Msg_Log (Array("1022" , "COPY"))
'-----------------------------------
'COPY  SI + CC ==> TRDUAV
Call CopMFil (Rep_TMP & "\WSI.tmp" ,Rep_TMP & "\WCC.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>
```

DSMS Update Before Database Update

This update is performed using, as input to 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 TRRP procedure generates transfer transactions. These have the same format as batch update transactions applicable in the Development Database by the UPDT procedure.

The transfer transactions may be generated on all the sets related to a Development Database, or on selected sets, based on the following criteria:

1. Transaction Set (required),
2. Target Session.

The 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 (by transaction set)

- Transaction Set code different from '**********'.
- No target session

TRRP only processes the transactions that belong to the selected Transaction Set. Since you have not selected a target session, transactions are generated for all the 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 parameters file (UY) once all the 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 parameters 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 Pac/transfer operations involve large volumes of transactions.

A Warning message will tell you when all sessions have been processed.

Generated transactions must then be used by the batch update procedure (UPDT) of the Development Database.

You may prefer to concatenate all TRRP subsequent outputs and run the UPDT procedure only once.

Execution by Transaction Set

- Transaction Set code different from '**********'
- Target session: 'nnnnT' or '*****'

TRRP only processes the transactions that belong to the selected Transaction Set.

- 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 (UY) once all the transactions have been generated for a given session.

Generated transactions must then be used by the Development Database update procedure (UPDT).

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.

Note: A specific attribute -- SESSION PROCESSED -- is automatically positioned in the parameters file (UY) once all transactions have been generated for a given session.

Generated transactions must then be used by the Development Database batch update procedure (UPDT).

Execution conditions

The Transfer file must exist (created by the TRPF procedure).

Result

Transfer transactions formatted for the Development Database batch update procedure (UPDT).

TRRP - User Input

User identification line (required)

Position	Length	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):
		llllllllll	Transaction Set code
		'**********'	All Transaction Sets
43	1		Formatting for UPDT
		'1'	Formatting
		' '	No formatting
44	1		Formatting for UPDP
		'1'	Formatting
		' '	No formatting

Sort Options line

Position	Length	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
7	1	' '	List by session
		'N'	No list by session

TRRP - Description of Steps

Preparation of extraction: PTUG60

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database index
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database data
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database users
PAC7AR	Database dir. : AR	Input	Development Database data
PAC7JT	Save dir. : TJ	Input	Sequential or compressed Journal
PAC7MB	User input	Input	User input
PAC7UY	User dir.: UY	Input	Reduced parameters
PAC7BM	Tmp dir. : WMB	Output	Extraction request for PACX
PAC7PJ	Tmp dir. : WPJ	Output	Temporary file
PAC7ET	User dir. : TRRPETG60	Report	Statistics on transfer
PAC7DD	User dir. : TRRPDDG60	Report	Check on user

Return codes:

- 0: No error
- 8: Critical error (detailed in PAC7DD)

Extraction: PACX

This step extracts the transactions according to the user input.

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7AN	Database dir. : AN	Input	Development Database index
PAC7AR	Database dir. : AR	Input	Development Database data
PAC7AY	Database dir. : AY	Input	Development Database extension data
PACGGN	Admin Database - Base dir. : AN	Input	Administration Database index
PACGGR	Admin Database - Base dir. : AR	Input	Administration Database data
PACGGU	Admin Database - Base dir. : GU	Input	Administration Database user file
PAC7PJ	Tmp dir. : WPJ	Input	Transactions selected by the Journal
PAC7MB	Tmp dir. : WMB	Input	User input
PAC7BM	Tmp dir. : WBM	Input/Output	User input
SYSEXT	Tmp dir. : WSY	Input/Output	Work file
PAC7MJ	Tmp dir. : WMJ	Input/Output	Journal transactions (EXPJ)
PAC7WD	Tmp dir. : WWD	Input/Output	Extracted transactions
PAC7MV	User dir. : MV	Output	Extracted transactions for UPDT
PAC7GY	User dir. : GY	Output	Extracted transactions for UPDP
PAC7IA	User dir. : TRRPIAPAC	Report	Overall program flow printout
PAC7DD	User dir. : TRRPDDPAC	Report	Printing of errors list on input transactions
PAC7EE	User dir. : TRRPEEPAC	Report	Extractions report
PAC7EP	User dir. : TRRPEPPAC	Report	Extractions report
PAC7EQ	User dir. : TRRPEQPAC	Report	Extractions report
PAC7EZ	User dir. : TRRPEZPAC	Report	Extractions report

Return codes:

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

Positioning the 'session processed' attribute: PTUG61

Code	Physical name	Type	Label
PAC7AE	System - Skel dir. : AE	Input	Error messages
PAC7AR	Database dir. : AR	Input	Development Database data
PAC7MB	User input	Input	User input
PAC7UY	User dir. : UY	Input/Output	Reduced parameters
PAC7ET	User dir. : TRRPETG61	Report	Transfer statistics

TRRP - Execution Script

```
' ------------------------------------------------------
'      VISUALAGE PACBASE
'
' ------------------------------------------------------
'       PAC/TRANSFER -
'       GENERATING THE TRANSFER TRANSACTIONS
' ------------------------------------------------------
'
' ONCE THE TRANSFER FILE HAS BEEN BUILT, THE TRRP
' PROCEDURE GENERATES TRANSFER TRANSACTIONS. THESE HAVE
' THE SAME FORMAT AS BATCH UPDATE TRANSACTIONS
' APPLICABLE BY THE UPDT PROCEDURE.
'
' INPUT :
'  -  USER IDENTIFICATION LINE (REQUIRED)
'     COL  2 : "*"
'     COL  3 : USERIDXX
'     COL 11 : PASSWORD
'     COL 22 : (5 CAR.) SELECTION OF TARGET SESSION(S)
'     COL 33 : (10 CAR.) SELECTION OF TRANSACTION SET(S)
'     COL 43 : "1"   UPDT FORMAT
'              " "   NO FORMAT
'     COL 44 : "1"   UPDP FORMAT
'              " "   NO FORMAT
'  -  COMMAND LINE :
'  COL 2  : "J"   LINE CODE
'  COL 4  : " "   CHRONOLOGICAL LIST
'           "N"   NO CHRONOLOGICAL LIST
'  COL 5  : " "   LIST BY USER
```

```
'           "N"   NO LIST BY USER
'  COL 6  : " "   LIST BY LIBRARY
'           "N"   NO LIST BY LIBRARY
'  COL 7  : " "   LIST BY SESSION
'           "N"   NO LIST BY SESSION
' ------------------------------------------------------
'
<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

Call Msg_Log (Array("1022" , "PTUG60"))
'-----------------------------------
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"
Call BvpEnv("PTUG60","PAC7BM",Rep_TMP & "\WMB.tmp")
Call BvpEnv("PTUG60","PAC7DD",Rep_USR & "\TRRPDDG60.txt")
Call BvpEnv("PTUG60","PAC7ET",Rep_USR & "\TRRPETG60.txt")
WshEnv("PAC7JT") = Rep_SAVE & "\TJ"
WshEnv("PAC7MB") = Fic_Input
Call BvpEnv("PTUG60","PAC7PJ",Rep_TMP & "\WPJ.tmp")
Call BvpEnv("PTUG60","PAC7UY",RepT_USR & "\UY")
Call RunCmdLog ("BVPTUG60")
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"
Call BvpEnv("PACX","PAC7DD",Rep_USR & "\TRRPDDPAC.txt")
Call BvpEnv("PACX","PAC7EE",Rep_USR & "\TRRPEEPAC.txt")
Call BvpEnv("PACX","PAC7EP",Rep_USR & "\TRRPEPPAC.txt")
Call BvpEnv("PACX","PAC7EQ",Rep_USR & "\TRRPEQPAC.txt")
Call BvpEnv("PACX","PAC7EZ",Rep_USR & "\TRRPEZPAC.txt")
Call BvpEnv("PACX","PAC7IA",Rep_USR & "\TRRPIAPAC.txt")
Call BvpEnv("PACX","PAC7MV",RepT_USR & "\MVTRRP.txt")
Call BvpEnv("PACX","PAC7BM",Rep_TMP & "\WBM.tmp")
```

```
Call BvpEnv("PACX","PAC7GY",RepT_USR & "\GYTRRP.txt")
Call BvpEnv("PACX","PAC7MB",Rep_TMP & "\WMB.tmp")
Call BvpEnv("PACX","PAC7MJ",Rep_TMP & "\WMJ.tmp")
WshEnv("PAC7MM") =  Rep_TMP & "\NULMM.tmp"
'PAC7MM not used, on default
Call BvpEnv("PACX","PAC7PJ",Rep_TMP & "\WPJ.tmp")
WshEnv("PAC7MR") =  Rep_TMP & "\NULMR.tmp"
'PAC7MR not used, on default
WshEnv("PAC7RE") =  Rep_TMP & "\NULRE.tmp"
'PAC7RE not used, on default
WshEnv("PAC7RM") =  Rep_TMP & "\NULRM.tmp"
'PAC7RM not used, on default
WshEnv("PAC7TE") =  Rep_TMP & "\NULTE.tmp"
'PAC7TE not used, on default
WshEnv("PAC7UE") =  Rep_TMP & "\NULUE.tmp"
'PAC7UE not used, on default
WshEnv("PAC7TD") =  Rep_TMP & "\NULTD.tmp"
'PAC7TD not used, on default
Call BvpEnv("PACX","PAC7WD",Rep_TMP & "\WWD.tmp")
Call BvpEnv("PACX","SYSEXT",Rep_TMP & "\WSY.tmp")
Call RunCmdLog ("BVPACX")
If Return = 4 Then
Call Msg_Log (Array("1030"))
End If
If Return = 8 Then
Call Msg_Log (Array("1057"))
End If
Call Err_Cod(Return , 0 , "PACX")

Call Msg_Log (Array("1022" , "PTUG61"))
'-----------------------------------
WshEnv("PAC7AE") = Rep_SKEL & "\AE"
WshEnv("PAC7AR") = Rep_BASE & "\AR"
Call BvpEnv("PTUG61","PAC7ET",Rep_USR & "\TRRPETG61.txt")
WshEnv("PAC7MB") = Fic_Input
Call BvpEnv("PTUG61","PAC7UY",RepT_USR & "\UY")
Call RunCmdLog ("BVPTUG61")
Call Err_Cod(Return , 0 , "PTUG61")


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 Development 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 Development Database).

Reinitialization of DSMS Environment

This processing in the DSMS Database resets the update authorizations on the selected source and target Changes as they were before the transfer operation.

This initial state is obtained by executing the DSMS update procedure (DUPT), using as input transactions the contents of the file resulting from the DSMS Environment Preparation procedure (TRDU).

ASCII Sort

ASCII Sort of User Parameters

PEAS - Introduction

The PEAS procedure sorts the user parameter backup file (PE) 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.

PEAS - Description of Steps

ASCII sort on PE file: PTU903

Code	Physical name	Type	Label
PAC7IN	User dir. : WIN	Input	Original user parameters
PAC7OU	User dir. : WOU	Output	User parameters sorted as an ASCII sequence

PEAS - Execution Script

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

```
<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


Call Msg_Log (Array("1022" , "PTU903"))
'-----------------------------------
Call BvpEnv("PTU903","PAC7IN",Rep_USR & "\WIN.txt")

Call BvpEnv("PTU903","PAC7OU",Rep_USR & "\WOU.txt")
Call RunCmdLog ("BVPTU903")

Call Err_Cod(Return , 0 , "ptu903")

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


</script>
</job>
```

ASCII Sort of Generation Requests

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

Code	Physical name	Type	Label
PAC7IN	User dir.: WIN	Input	Original generation requests
PAC7OU	User dir.: WOU	Output	Generation requests sorted as an ASCII sequence

PGAS - Execution Script

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


Call Msg_Log (Array("1022" , "PTU906"))
'-----------------------------------
Call BvpEnv("PTU906","PAC7IN",Rep_USR & "\WIN.txt")

Call BvpEnv("PTU906","PAC7OU",Rep_USR & "\WOU.txt")
Call RunCmdLog ("BVPTU906")

Call Err_Cod(Return , 0 , "ptu906")

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


</script>
</job>
```

ASCII Sort of Environments

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

Code	Physical name	Type	Label
PAC7IN	User dir.: WIN	Input	Original environments

Code	Physical name	Type	Label
PAC7OU	User dir.: WOU	Output	Environments sorted as an ASCII sequence

PPAS - Execution Script

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


Call Msg_Log (Array("1022" , "PTU907"))
'-----------------------------------
Call BvpEnv("PTU907","PAC7IN",Rep_USR & "\WIN.txt")

Call BvpEnv("PTU907","PAC7OU",Rep_USR & "\WOU.txt")
Call RunCmdLog ("BVPTU907")

Call Err_Cod(Return , 0 , "ptu907")

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

</script>
</job>
```

Part Number: DELNT002357A - 7942

Printed in USA