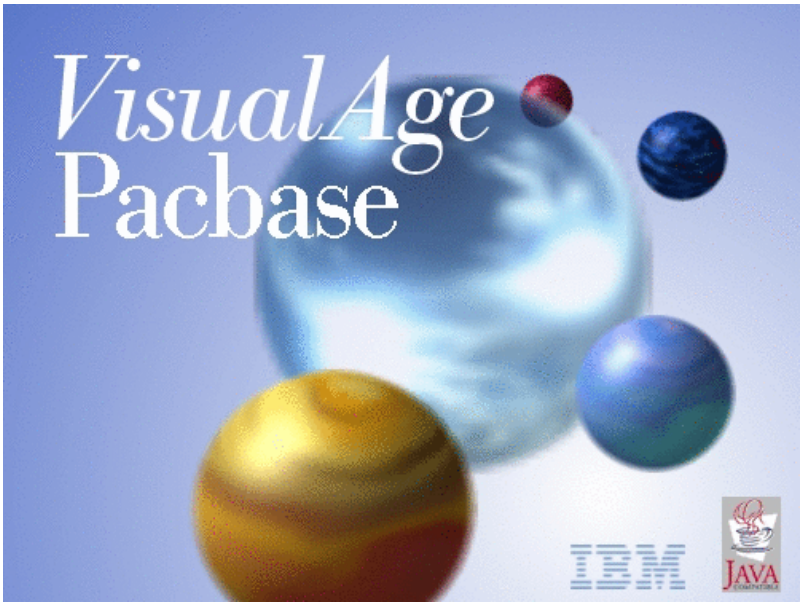
VisualAge Pacbase

PACBASE ACCESS FACILITY

Version 3.5



VisualAge Pacbase

PACBASE ACCESS FACILITY

Version 3.5

Note

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

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&context=SSEP67&uid=swg27005477

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

First Edition (January 2007)

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

Contents

Notices

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

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

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

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

Trademarks

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

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

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

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

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

Chapter 1. The Pacbase Access Facility (PAF) function

Introduction

The Pacbase Access Facility (PAF) function enables you to extract information from the VisualAge Pacbase Development and Administration Databases via SQL statements. This extracted information can then be used to update the VA Pac Databases.

RELATIONAL VIEW OF THE DATABASE

The PAF Function provides a relational description of the standard VisualAge Pacbase metamodel. This type of description is a prerequisite for the formulation of SQL queries.

A relational model is a tabular structure of data organized in columns. Relations between objects (data bases, table spaces, tables, views, or indexes - but not the data itself) are correspondences between the table columns.

In the PAF tabular model:

- Each entity is described by a set of tables.
- Information about an entity definition or description is described by specific columns.
- Cross-references are described in two ways:
  - By direct relations between two columns of different Tables,
  - By 'virtual' tables, whose columns identify the two tables to be related and represent the cross-references.

>>>> See next subchapter for a detailed description of the PAF tabular model.

The Structured Query Language (SQL) is the standard query language used with Relational Databases. Its syntax is similar to that of the English language.

This language is used to formulate queries, retrieve selected data, and update the database.

An SQL query is defined by a SELECT statement (see subchapter 'Syntax of the SQL-PAF Language').

The tables in a Relational Database are composed of rows (equivalent to records of a file).

An SQL query defines a subset of the information contained in a database without specifying when or how this information was obtained and processed.

In order to integrate SQL with the procedural languages used in business-oriented computing, the concept of the cursor has been associated with an SQL query.

A cursor makes the extracted rows available to a PAF user program. The cursor is defined and declared with a DECLARE CURSOR statement, which includes a SELECT statement that identifies the extracted rows. When a cursor is used, the program can retrieve each extracted row sequentially: the cursor must be opened (with an OPEN statement) before any rows are retrieved. A FETCH statement is used to retrieve the cursor's current row, and can be executed repeatedly until all rows are retrieved. Then, the cursor must be closed with a CLOSE statement.

Once the cursor has been defined, the statements make the cursor similar to a file; the FETCH statement is the equivalent of a READ statement in COBOL.

STANDARD SQL QUERIES

With the Pacbase Access Facility (PAF), the Development and the Administration Database can be accessed via the Structured Query Language (SQL), which is commonly used for Relational Databases.

From the tabular model, described in the next subchapter, accesses are performed by the declaration and use of SQL cursors.

Each cursor is associated with a VA Pac context (Library and Database Session). Several cursors can be processed at one time in order to access information pertaining to several VisualAge Pacbase cross-references.

Queries can be written directly in COBOL, described in all batch or online user programs, or generated by VisualAge Pacbase.

In all cases, the query is processed by the PAF Translator program (before the COBOL compilation), which translates the SQL statements into CALL instructions used by the PAF Extraction sub-program.

THE PAF EXTRACTION SUB-PROGRAM

This sub-program accesses and extracts data from the Development and Administration Databases. It retrieves the internal parameters built by the PAF Translator Program in order to perform the requested data extraction.

Extracted data is transmitted to the PAF user program in the Communication Area generated by the PAF Translator Program (COBOL Communication Area in the WORKING-STORAGE SECTION).

The PAF tabular model

INTRODUCTION

The PAF relational model is a tabular structure of data organized in columns. These columns are defined by VA Pac Data Elements and the tables are associated with Segments or extension Meta Entities.

All of the entities which make up the PAF tabular model are provided in the form of batch transactions which can be used to update a VA Pac Library:

- Data Element Definitions, which represent the Columns in the PAF Tables,
- Data Structure Definitions, with one Data Structure per VA Pac entity type,
- Segment Definitions and extension Meta Entities which represent the PAF Tables,
- Descriptions of these Segments (the Call of Data Elements) and extension Meta Entities.

All of these entities include the keyword 'PAF'.

CODING RULES AND DESCRIPTION OF TABLES

Tables are generally assigned a 6 to 10-character mnemonic code. The first three characters identify the entity concerned.

This code must be specified in the cursor declaration:

SELECT * FROM <code-table>.

NOTE: The Documentation of the PAF Tables is available at the following address:

www-1.ibm.com/support/docview.wss?rs=37&uid=swg27005477.

Click on the 'PAF module' choice. The links you find there lead you to the lists of the PAF tables of all the VA Pac entities, sorted by functional unit:

- the entities of the standard VA Pac metamodel,
- the entities of the eBusiness metamodel,
- the entities managed in the Administration database and the Administration management entities linked to a Development database,
- the Methodology entities.

A cumulative index of all the tables for all the entities is also available at this address. This index enables you to branch directly to the description of a selected table.

VIRTUAL TABLES

Virtual tables represent part of an entity's cross-references. They are coded with an 'X' inserted between two Entity Table codes for VA Pac.

Virtual tables have been created for cross-references which cannot be defined by a simple SQL elementary condition.

EXAMPLE: Data Element calls on Programs' Procedural Code (-P) lines.

This cross-reference is built when a Data Element code has been entered in the OPERANDS or in the CONDITION FOR EXECUTION field on a '-P' line in a Program.

In addition, both fields may contain other information, such as another Operand, for example. The SQL syntax does not allow for the description of such a cross-reference using just the columns of the PGMPRC Table, which describes the Program's Procedural Code lines.

Consequently, a specific table has to be created in order to implement that cross-reference. This Table is coded DELXPGMPRC and contains the Columns which identify both the Data Element and Program entities: Data Element Code, Program Code, Function Code, Sub-Function Code, and Line Number. It also includes all the columns in the PGMPRC Table.

NOTE: A Virtual Table contains at least the Columns identifying two entities.

VA Pac Metamodel Entities

In the standard VA PAC metamodel, the first three characters of the table code which identify the entity are:

```
Database                    DBT
Library                     LIB
Data Element                DEL
Database Block              DBD
Dialogue                    DLG
Data Structure              DST
Input Aid                   PIA
Merise
         Model F.I.C.       FIC
         Model Object       OBJ
         Model Relationship REL
Program                     PGM
Report                      RPT
Screen                      SCR
```

```
Segment                     SEG
Volume                      VOL
Client Meta-Entity          CME
Extension Meta-Entity       EME
Client User Relation        CRL
Extension User Relation     ERL
Client User Entity          $TT
Extension User Entity       YTT
```

The next three characters identify the Entity screen.

```
Definition                  DEF
Comment                     DOC
Assigned Text               TXT
Called in Volume Desc.      VOL
Description                 DSC
User Entity Description     Dxx
Hierarchical Description    DSH
Codasyl Description         DSC
Description - Heading (M.E.)DSE
Relational Description      DSR
Error labels                ERR
Generation elements         GEN
Parameters                  OPT
Entity calling              CAL
Call of Data Elements       DEL
Call of Programs            PGM
Call of Segments            SEG
Call of Data Structures     DST
Call of Objects             OBJ
Work Areas                  WSS
Beginning Insertions        BEG
Procedural Code             PRC
Procedural Code 8           PR8
Call of Macro-Structures    PMS
Pure COBOL Source Code      COB
Source Code (REVERSE)       SRC
Dialogue Complement         CMP
Sub-Schema/System Labels    SSL
Report Layouts              LBL
Report Structures           STR
Report Categories           CAT
```

EXAMPLES

DELDEF Data Element Definitions

TXTDSC Text Descriptions

SEGDEL Segments' Call of Elements

SPECIAL TABLES

```
LCKENT      List of Locked Entities
FROSES      List of Frozen Sessions
TIMESTAMP   List of timestamps
KYWDEF      Keyword thesaurus
WORDSEARCH  Wordsearch
```

TABLE COLUMN CODING RULES

A Table Column is a VA Pac Data Element and has a 6-character code.

- The first character identifies the NATURE of the data: a Code, a Label, a Type, an Option, etc.
- The next three characters indicate the OBJECT or FOCUS of the data: a Record, Structure, Function, etc.
- The last two characters indicate a QUALIFIER: a Parent (for a Data Element), a Value (for a type or option), an Input, Internal or Output Format, etc.

The first four characters have fixed values which are listed below. The last two characters are determined by the column they describe.

EXAMPLE: The 'CL' suffix may have several meanings:

1. 'CLASS': for the ODELCL column, called 'CLASS (ALPHA/NUMERIC)',
2. 'COLOR': for the CATTCL column, called 'COLOR ATTRIBUTE-LABEL'.

DATA NATURE CODING RULES

```
C: Identifying code or number
L: Label, clear name
E: External name
T: Type
O: Option
N: Number (meaning rank or level)
F: Format, length
A: Address, position
D: Documentation, description, comments
V: Value, contents
```

OBJECT/FOCUS CODING RULES

```
LIN: Line
REC: Record
TRA: Transaction
RES: Record Structure
SSS: Sub-Schema or Sub-System
LBL: Label
STR: Structure
CAT: Category
FCT: Function
SFC: Sub-Function
CND: Condition
```

```
ATT: Data Element Attribute
LAN: Generation Language
CCF: Control Cards in Front
CCB: Control Cards in Back
SEC: Section (COBOL Program and WorkStation-managed Text)
PAR: Paragraph
DIV: Division
SET: Set
CHA: Level 1 (Volume Section)
SCH: Level 2 (Volume Section)
ENT: Entity
SEN: Sub-Entity
SES: Session
USR: User
LCK: Entity Update Lock
DSC: Description
SEL: Selection
VER: Version
KWD: Keyword
```

Each VA Pac entity type is considered to be an Object and, therefore, is coded in the same way as the first three characters of PAF Table Codes (DEL, SEG, etc.) except for Extension User Relations which are coded 'ERL' in tables and 'RLX' in columns.

EXCEPTION: Columns of WorkStation-Specific entities

These rules do not apply to entities managed by the WorkStation. These columns are special Data Elements dedicated to the Description of the WorkStation-specific Meta Entities. The same applies to entities defined by the Extension Meta Entities (Administration and eBusiness).

Methodology Entities

Tables which describe the VA Pac WorkStation follow specific coding rules.

The VA Pac WorkStation entities are described and stored as Meta-Entities in the Specifications database.

The first three characters identify the Meta-Entity.

The next three characters identify the ME screen: DEF, DOC, TXT, APP, and Dxx (xx identifies the Description of the associated Meta Entity.

EXAMPLE: The code of the Table which describes a CDM Definition is $1SDEF (Merise).

To edit the type codes of the Meta-Entities dedicated to the WorkStation and the identifiers of the descriptions specific to the current methodology, use the PCM command in the GENERATION AND PRINT COMMANDS screen (CHOICE: GP).

The code of the methodology is entered on one character in the ENTITY field.

The list of values for the various methodologies is the following:

M for Merise

D for YSM

Y for Yourdon

A for SSADM

O for OMT

F for IFW

Administration and Management Entities

Administration Entities

For these entities managed in the administration workspace (and which are common to several databases), the first three characters of the table code which identify the entity are:

Database	YAB
Access key	YAK
Command line	YAF
Pactransfer	YAR
Parameter	YAT
Procedure	YAP
Profile	YAV
Security	YAS
User	YAU
Endevor	YEN

Management Entities

For these entities managed in the development database, the first three characters of the table code which identify the entity are:

Extraction Master Path	Y7E
Quality rules Definition	Y5Q
Generation-print commands	YG1
Environment	YC1

eBusiness Entities

In the eBusiness model, the first three characters of the table code which identify the entity are:

Application	YCS
Elementary Component	YCE
Folder	YDO
Message	YMS
Communications Monitor	YMC

Operation	YOP
Part	YPT
Service	YSV
SOAP Binding	YSB
Initialization Server	YSI
Logical View	YVL

Chapter 2. Implementation in User Programs

Introduction

THE USER PROGRAM

PAF is implemented through an online or batch user program, either written directly in COBOL or generated by VA Pac. The PAF Extraction subprogram generates all the accesses to the Specifications Database.

In the user's program, the cursor(s) must first be declared via a DECLARE CURSOR statement in order to access the desired tables. For each declared cursor, the sequence of statements is as follows:

CONNECT: Connect Cursor to a Visualge Pacbase context (User, Library, and Database Session),

OPEN: Open Cursor, i.e. extraction from the Specifications Database,

FETCH: Sequentially retrieve extracted rows,

CLOSE: Close Cursor.

The SET statement allows for the dynamic modification of the PAF Translator Program's operating parameters.

The INIT and QUIT statements perform technical initialization and termination operations according to the extraction mode and the hardware in use (e.g. File OPEN/CLOSE in batch mode). They are not required in online mode.

THE PAF TRANSLATOR

SQL commands are inserted into a PAF user program and are translated into COBOL instructions before the COBOL compilation.

The PAF Translator transforms SQL statements into comment lines which precede the translated COBOL instructions.

The SQL-PAF DECLARE statement is translated into a declaration. Other SQL commands are translated into CALLs of the Extraction subprogram, except for the SET statement, which is not an SQL command and has a very special usage.

For more information, please refer to paragraph 'The Set Statement' in subchapter 'Syntax of the SQL-PAF Language'.

The PAF Translator is parameterized by a comment line inserted into the PAF user program following the IDENTIFICATION DIVISION line. This parameter line, the description of which appears on the next page, is automatically generated if the program is developed with VA Pac and if the 'EXP' operator is used on a Procedural Code (-P) line of the program.

For more details, please refer to subchapter 'PAF Implementation Under VisualAge Pacbase'.

The PAF Translator parameter line is formatted as follows:

COLUMN	LENGTH	CONTENT
1	6	Cobol line number
7	1	* for Cobol comment line
8	5	Execution mode: batch or online
14	4	Fixed label
19	3	Library code
23	5	Session number (& version)
29	2	Generation variant(s) (Cobol & Map)
32	3	Fixed label
36	1	Database language code (A or F)
38	3	Batch program skeleton, online program skeleton, Cobol program skeleton
42	1	Skeleton language (A or F)
44	6	'Single' or 'double' quotes delimiter

The Execution Mode is used to distinguish between batch and online. The Execution Mode that VA Pac takes into account depends on the generator implemented on site. The Execution Mode allows the PAF Translator to declare, as appropriate, the work areas specific to online and to generate the calls to the Extraction subprogram.

The Generation Variant taken into account by VA Pac depends on the one specified on the Program Definition screen. It is used to adapt the generated syntax according to the compiler.

The String Delimiter which VA Pac takes into account depends on what was specified on the Library Definition. It allows PAF Translator to recognize the string delimiter for both generation and source analysis.

The Library and Session parameters allow the PAF Translator to connect to the appropriate VA Pac Database when it is processing a Cursor dealing with a User Entity: the columns associated with User Entity depend on the description of the corresponding Meta-Entity. Therefore, the PAF Translator has to read this Meta-Entity in the Library and Session where it is described in order to validate the SQL query.

The PAF Translator is a bilingual program. The first specified Language Code applies to error messages generated by the Translator. The second refers to the language used in the mnemonic coding of the Tables and Columns which describe the VisualAge Pacbase Database.

This implementation of two language codes (English and French) allows a site to generate programs for another site using a different language code. The values taken by VA Pac for these two language codes are those of the AE and SG files (error messages and generation skeleton).

The line 2 generated by the VA Pac generator is read by the PAF generator, but is not reproduced in the translated source.

The SET statement can be used to easily modify these parameters anytime during the actual translation process. For example, it may be necessary to change the Library Code one or more times. However, in the case of an on-line program containing the INSERT and FETCHER statements (without DECLARE CURSOR), the SET statement must come before the previous statements for the initialization of the area (e.g., writing the statement in Working Storage Section).

THE PAF EXTRACTOR

The extractor subprogram manages accesses to the VisualAge Pacbase development and administration Databases.

It retrieves the internal parameters built by the PAF Translator and performs the selected data extraction as follows:

- When a CONNECT statement is issued, the Extractor establishes the user's connection, for the specified Cursor, to the VA Pac Database (validates access authorization to Library and Database Session).
- When an OPEN statement is issued, the Extractor accesses the VA Pac Database, and stores the extracted rows in an temporary Work File. The

number of extracted rows may be parameterized for each Cursor (SIZE parameter within the CONNECT Statement).

- When a FETCH statement is issued, the Extractor retrieves the extracted rows, one by one, from the Temporary Work File and sends them to the user program Communication Area generated by the PAF Translator.

  Refer to subchapter 'Database Access Optimization' which provides a detailed explanation of the mechanism used by the Extractor subprogram to manage the OPEN and FETCH statements.
- When a CLOSE statement is issued, the specified Cursor is closed.

DAF extractor is also used in the DUPD procedure, which updates the DSMS database using the DAF Tables sequential file.

For more information, refer to the DSMS Operations Manual, chapter 'Batch Updates from DAF Tables (DUPD)'.

THE TEMPORARY WORKFILE

The purpose of the temporary Workfile is to store the rows extracted from the development or administration databases after an OPEN statement is issued.

Extracted rows are retrieved one by one from the temporary Workfile for each FETCH statement. The maximum number of extracted rows may be parameterized for each Cursor (via the SIZE parameter in the CONNECT statement).

For more details, refer to Subchapters 'Syntax of the SQL-PAF Language' and 'Database Access Optimization'.

The temporary Workfile also contains technical parameters used by the PAF Extractor subprogram for Cursor management.

PHYSICAL DESCRIPTION

The temporary Workfile is an indexed sequential file with a variable format. The stored rows are linked to a user and a SQL cursor.

The key structure is slightly different in batch and online mode.

When this file is created for a batch job, its access key is composed of:

- A Cursor code,
- A Structure code,
- A Record number.

When this file is created for online use, it is used by all PAF applications and users, and its access key is composed of:

- A Conversation identifier,
- A Cursor code,
- A Structure code,
- A Record number.

Syntax of the SQL-PAF Language

GENERAL INFORMATION

To access the development and administration databases through the PAF function, you have to declare, and then use, SQL Cursors.

For each table to be accessed, you must declare a Cursor in the WORKING-STORAGE SECTION (DECLARE CURSOR statement). In the PROCEDURE DIVISION, the sequence of statements associated with a given Cursor is as follows:

CONNECT, OPEN, FETCH, and CLOSE.

OPEN, FETCH and CLOSE are standard SQL statements, while the CONNECT statement is specific to the PAF function. All four of these statements are designated as Cursor operation statements in SQL-PAF syntax.

A PAF user program can use up to 100 Cursors.

The INIT (initialization) and QUIT (termination) statements, which are independent of Cursors, must be issued, respectively, before and after any Cursor operation statements (compulsory in Batch mode).

NOTE: All SQL-PAF sentences must be coded starting in COBOL column 12, must begin with EXEC PAF, and must end with END-EXEC.

CURSOR DECLARATION

An SQL Cursor is declared in the WORKING-STORAGE SECTION of the PAF user program by means of a DECLARE CURSOR statement.

In the DECLARE statement, the following keywords should be noted:

SELECT, FROM, WHERE, AND, and OR.

The syntax of the DECLARE CURSOR statement is as follows (values between parentheses are optional):

```
EXEC PAF DECLARE <cursor-code> CURSOR FOR
SELECT * FROM <table-code> (WHERE <condition(s)>)
END-EXEC
```

where:

- <cursor-code> is the four-character cursor identifier,
- SELECT here applies to the whole table, and is used to retrieve all table columns; in other words, columns cannot be selected individually. Thus, the syntax is always SELECT *.
- FROM cannot be used with a JOIN clause, and is therefore followed by just one table code.
- <table-code> identifies the PAF Table.
- WHERE does not allow SUBSELECTs.
- <condition(s)>: Each condition applies to a table column and is indicated in parentheses.

  Several conditions may be linked using the logical 'AND' and 'OR' operators. The total number of elementary conditions is limited to 50.

  A condition is formatted as follows:

  COLUMN OPERATOR OPERAND

  where:

  – COLUMN = column code
  – OPERATOR may have the following values:

```
=   : equals
>   : is greater than
>=  : is greater than or equal to
<   : is less than
<=  : is less than or equal to
<>  : is different from
```

- OPERAND is either:

```
. Another column of the table,
. A COBOL constant,
. A numeric or alphanumeric constant,
. A COBOL variable of the PAF user program.
```

NOTE: Alphanumeric constants cannot exceed 60 characters. If this length is insufficient, an initialized COBOL variable can be used instead.

Numeric constants can only be unsigned integer constants and cannot exceed 18 digits. The PAF Translator does not validate the declaration of COBOL variables used as operands. Subscripted COBOL variables cannot be created. As far as limited conditions on user entities are concerned, the inferior value has to be placed at the 1st place and the the superior value has to be placed next :

WHERE: CUEO > ..... AND CUEO < ..... .

Limitations related to the use of the SELECT clause are not restrictive, since the ability to manage several cursors at the same time makes up for the inconvenience of mono-table accesses. Furthermore, coding a very complicated SQL query is often a tricky matter. The purpose of the PAF Function is not to provide a comprehensive SQL interface, but to allow its users to access any data contained in the development and administration databases. Data is accessed at the PAF user program level.

For more information, please refer to subchapter 'Embedded PAF Cursors' in this chapter.

CURSOR MANAGEMENT

Cursor operation statements are written in the PROCEDURE DIVISION of a PAF user program.

CONNECT: This is the first statement to be issued. It performs the connection to a VisualAge Pacbase development or administration database. This context can be modified as many times as needed.

The syntax of the CONNECT statement is as follows:

```
EXEC PAF CONNECT <cursor-code> TO
    USER = <user-code>
   The user code must have the authorization to access the
    administration database.
    PASS = <password>
    LIB = <library-code>
        = <***> in the administration workspace.
    SESSION = <session-number-and-version>
        = <0001Z> in the administration workspace.
    NET = <sub-network-option>
    SIZE = <maximum-number-of-rows>
    IDENT = <transaction-code>
    BASE = <VA Pac-Database-code>
         =no BASE parameter in the administration workspace.
END-EXEC
```

The parameters to the right of the equal sign '=', which are described below, must be either literals or COBOL variables.

USER: VisualAge Pacbase user code

PASS: VisualAge Pacbase user password

LIB: VisualAge Pacbase library code

SESSION: VisualAge Pacbase session number and version

NET: VisualAge Pacbase Database sub-network option:

I, C, A, U, <, >, and Z

SIZE: Maximum number of rows stored in the temporary Work File

IDENT: Conversation transaction code

BASE: 4-character Database code

All of the parameters in the CONNECT statement are required for the first cursor connection, except the BASE parameter in the administration workspace. The CONNECT statement is used to assign values to the parameters associated with the cursor during extraction. However, the user may modify certain parameters during a subsequent connection (for example, to use the same request in a different VA Pac context).

It is always possible to reconnect a cursor (for example to explore another sub-network with the same query). In order to request another VA Pac connection, the user only has to enter the parameters for the new connection and does not have to enter them in the order specified above.

```
EXEC PAF CONNECT <cursor-code> TO
NET = <sub-network-option>
LIB = <library-code>
END-EXEC
```

NOTES ON THE 'IDENT' AND 'BASE' PARAMETERS

These two parameters are only used in on-line PAF user programs.

The IDENT parameter is a 25-character field which identifies the online conversation using the PAF function. It is also part of the access key in the temporary Work File. Its value depends on the monitor in use. For example, on CICS, it is advised to set this parameter with the DFHTERMID variable supplied by CICS. For complete information on the IDENT values, refer to 'The 'IDENT' Parameter' subchapter.

The BASE parameter contains the four-character database code and is used to access a VA Pac development database and the PAF Temporary Work File.

This parameter is not required to access the administration database. This parameter is used only with IBM hardware running on CICS).

NOTES ON THE 'SIZE' PARAMETER

The maximum number of extracted rows in the temporary Workfile may be exceeded in the case of the following tables:

- Segment Descriptions (SEGDEL),
- Entity Comments (entDOC) or Entity Decription Comments (entdscDOC) in which Parameterized Input Aids are called.

For these three types of tables, the extraction proceeds until the end of the Segment or PIA description. It stops when the SIZE limit (max. no. of rows) is reached or exceeded.

The SIZE parameter value must be greater than zero. Otherwise, the PAF Extractor subprogram automatically sets it to '1'.

OPEN: When this statement is issued, the Extractor Subprogram accesses the VA Pac Database and writes, in the temporary Workfile, the rows selected (and extracted) according to the cursor declaration.

A cursor can be opened if it is already connected and if it has not yet been opened.

The syntax of the OPEN statement is:

```
EXEC PAF OPEN <cursor-code> END-EXEC
```

FETCH: When this statement is issued, the Extractor subprogram sends the current retrieved row from the Intermediate work file to the PAF user program.

The cursor must be opened and there may be as many FETCH statements as necessary.

The syntax of the FETCH statement is:

```
EXEC PAF FETCH <cursor-code> END-EXEC
```

NOTE: The standard syntax of the FETCH statement includes the keyword 'INTO', which allows each selected column to be associated with a COBOL field.

The PAF-SQL extractor extracts ALL columns into a Communication Area, generated in the WORKING-STORAGE SECTION when a DECLARE statement is issued.

Thus, the keyword INTO (followed by the list of target COBOL fields) is not used in the SQL-PAF FETCH statement.

CLOSE: Only an opened cursor can be closed: a CLOSE statement can be issued only after an OPEN or a FETCH statement.

The syntax of the CLOSE statement is:

```
EXEC PAF CLOSE <cursor-code> END-EXEC
```

INSERT: INSERT enables to insert data in the temporary workfile (for example, a PAF extraction). This data must be stored initially in the pref-PUFENR field of pref-PAFCOM - 'pref' being a prefix of 1 to 4 character-long.

For the update, all the fields of the record to be updated must be valued: if not, these will be blanked out (unlike the UPDT and UPDP batch procedures).

The possible different values of the transaction code are: 'C' for a creation, 'M' for a modification , 'A' for a deletion and 'B' for a multiple deletion. This value is stored in the pref-PUFMVT field of pref-PAFCOM.

NOTE: At the time of an INSERT, the workfile may already contain '70'-type rows with the same identifier provided by previous transactions. Consequently, before inserting new data in the workfile, these records are re-read in order to determine the number of the statement which will be assigned when the next row is inserted (pref-NUMENR field of pref-PAFCOM).

CALPUF: This statement changes itself into a call to the PUF subprogram.

FETCHER: It returns, one by one, the errors which have been stored in the intermediate file by the PUF subprogram in the pref-PUFERR field of pref-PAFCOM. The user program is the one who must contain the restitution loop.

INSERT, CALPUF and FETCHER are independent of the cursors and exist in online mode only.

THE PAF TRANSLATOR'S PARAMETERS AND THE SET STATEMENT

SET:

The SET statement is used to modify a number of parameters used by the PAF Translator.

These parameters are initialized by the second line (a COMMENT line following the IDENTIFICATION DIVISION) in a PAF user program. This line is described in this chapter, in the 'Introduction' Subchapter, Paragraph 'The PAF Translator'.

After the specified parameters are actually updated, the PAF Translator sends the SET statement as comments to the translated PAF user program. The SET statement is the only SQL-PAF statement which is not translated into COBOL.

To generate the fields required by PUF, you must declare the SET statement at the very beginning of the Working Storage Section, in the first EXEC PAF. In this EXEC PAF SET, you can use the UPDATE, BASPUF and IDPUF parameters, which are specific to PUF.

You can also use the SET statement in the Procedure Division, provided you have first declared SET at the very beginning of the Working Storage Section, in the first EXEC PAF. In this case you can use it for the SESSION, LIB... parameters which are dynamically modified.

The syntax of the SET statement is as follows:

```
EXEC PAF SET
    LIB = <Library code>
    DELIM = <delimiter>
    TYPE = <generation variant(s)>
    MODE = <execution mode>
    LINK = <calling mode>
    SESSION = <session-number-and-version>
    USRENT = <UE parm>
    CTXCOL = <CTX parm (context)>
    UPDATE = <pref>
    BASPUF = <base-PUF>
    IDPUFL = <ident-PUF>
END-EXEC
```

The syntax of the SET statement is similar to that of the CONNECT statement. In particular, it is not necessary to give new values to the LIB, DELIM, TYPE, and MODE parameters and to enter them in the order given above.

THE PAF TRANSLATOR PARAMETERS

These parameters are not, unlike the CONNECT parameter, constant values or COBOL variables. They are purely alphanumeric and do not require a string delimiter.

EXAMPLE: In order to modify the LIB parameter, using the Library code 'PFA', the syntax would be as follows:

EXEC PAF SET LIB = PFA END-EXEC

Description of parameters

LIB: is a 3-character parameter. Its default value is the Library from which the PAF user program is generated.

It is used to declare a cursor for a table of User Entities (UEs) described by Meta-Entities which are not defined in the Library from which the PAF user program is generated.

DELIM: is a six-character parameter. It is used to modify the delimiter using the following values:

SINGLE: for the sinqle quote (’),

DOUBLE: for the double quote (″).

TYPE: is a two-character parameter. It is used to modify the variants VA PAC TYPE OF TP MONITOR and MAP TO GENERATE/TYPE OF COBOL TO GENERATE.

EXAMPLE: When a PAF user program is written and generated with the VA Pac Batch Systems Development component in an IBM CICS/MVS environment (Variants '00' or 'X0'), in order to obtain an On-Line application to be executed in an IBM IMS/VS environment, the TYPE parameter value needs to be changed to '01' or 'X1' ('00' is automatically generated by the PAF Translator).

Refer to the 'On-Line Systems Development' and 'Batch Applications' Manuals for a list of the values of these variants.

MODE: is a five-character parameter. It is used to modify the execution mode for VA Pac users who are using the Batch generator to generate online programs (value of MODE = 'TP').

LINK: is a six-character parameter. It is used to specify the calling mode of the extractor (STATIC or DYNAM) for static or dynamic calls, respectively. For CICS, this parameter is ignored and the translator generates a LINK statement for the call to the extractor.

SESSION: is a five-character parameter. It is used to modify the session. Its default value is the session in which the PAF user program is generated.

USRENT: (1 to 5 characters). This parameter is used to modify the code of the 'keywords' column in the UE definition table, in order to avoid the problem of codes conflicting with the user Data Elements called in the description of the associated Meta Entity. The default code of this column is WUEO and specifies the explicit keywords in this table. Apart from this one, each $TTDxx table is made of specific columns which correspond to the Data Elements called in the -CExx of the associated Meta Entity. The USRENT parameter replaces the 'UEO' string in the column code:

USRENT = XXX renames the column as WXXX.

CTXCOL: (1 to 5 characters). This parameter is used to modify the codes of the context columns of every PAF table, in order to avoid the problem of conflicting codes in the definition or description tables with user Data Elements called in the description of the associated Meta Entity. The default codes of these columns are LCTX, SCTX, TCTX and for the Library code, session number and hierarchical indicator, respectively (see chapter 'Implementation in User Programs', subchapter 'The Translated User Program').

The CTXCOL parameter value replaces the 'CTX' string in each of these column codes:

CTXCOL = YYY renames these columns as LYYY, SYYY, TYYY and NYYY, respectively.

It is important that the USRENT and CTXCOL parameters are NOT assigned the same values, in order to avoid additional conflicts.

The 'UPDATE = pref' clause

Generates the description of the two workings which will be used at the time of the calls to the PAF extractor (pref-PAFCOM) or to the PUF program (pref-PUFCOM) with the help of the INSERT, CALPUF and FETCHER statements - 'pref' being a prefix of 1 to 4 characters. This clause can only be used in the Working Storage Section, in the SET statement.

The 'BASPUF = base-PFU' and 'IDPUF = ident-PUF' statements

The PUF statements are independent of PAF cursors. The PUF- specific identifiers must be declared in order to give access to the workfile.

- 'base-PUF': VA Pac transaction code (4 char.)

- 'ident-PUF': identifier (1-25 chars.) These clauses can only be used in the Working Storage Section, in the SET statement.

For the last two clauses, parameters on the right side of '=' may be constants (indicated between quotes) or COBOL variables (for CONNECT).

NOTE: SET does not allow to modify the prefix of the Programs called. Indeed, the value is fixed and equal to 'BVP'.

TECHNICAL INITIALIZATION AND TERMINATION

The INIT and QUIT statements allow the Extractor to perform technical initialization and termination operations which depend on the Operating System in use. As a result, both statements must be issued, respectively, before and after other SQL-PAF statements.

The syntax of the INIT statement is:

EXEC PAF INIT END-EXEC

The syntax of the QUIT statement is:

EXEC PAF QUIT END-EXEC

NOTE: INIT and QUIT in batch mode

If the PAF user program calls subprograms, these statements may be executed only once (in the calling program) but they must be written in each subprogram which contains PAF statements.

Database Access Optimization

THE WHERE CLAUSE

Using the WHERE clause improves overall performance by optimizing Database accesses and reducing the volume of extracted data stored in the SYSPAF intermediate file (only relevant data need be extracted).

- Optimizing Database Accesses:

A Table access will prove more or less efficient according to the WHERE clause utilization.

EXAMPLE: Access to the SEGDEL Table

- SELECT * FROM SEGDEL WHERE CDEL = 'cccccc'

  Data Element's cross-references to Segments

  Equivalent online choice: EccccccXS
- SELECT * FROM SEGDEL WHERE CSEG = 'ssss'

  Direct access to the Segment's Description

  Equivalent online choice: SssssCE
- SELECT * FROM SEGDEL WHERE CSEG > 'ssss'

  Prior access to the List of Segments, then -- for each Segment -- an access to its Description is performed.

```
Equivalent online choice: LCSssss, S____CE
```

- Decrease in Volume of Extracted Data (stored in the SYSPAF Temporary File):
  - Either select all the programs and then keep only those whose Column code TPGMNA = M,
  - Or select in a WHERE clause only the programs whose Column code TPGMNA = M.

    Extraction will obviously be quicker with the second solution.

CONCLUSION: It is highly recommended to use the WHERE clause and to use it with a maximum of restrictive paramaters.

THE SIZE PARAMETER

In order to optimize accesses to the VA Pac database, the PAF Extractor reads several rows in advance. Advance reading avoids a systematic Read each time a row is fetched; systematic Reads cause systematic resets (START) in the VA Pac Index (AN) file.

Advance reading may be seen as the logical equivalent of Input/Output BUFFERs used by File Access Methods implemented with any Operating System.

For each cursor, the user can parameterize the number of rows read in advance by using the SIZE parameter in the PAF function's CONNECT statement.

MANAGEMENT OF ROWS READ BY THE PAF EXTRACTOR

The PAF Extractor reads the Database and validates the row that has been read. If the result is valid, the row is stored in the temporary Workfile.

This process is repeated as long as the number of stored rows is less than the value of the SIZE parameter.

Processing ends when the Extractor READ function detects the last row for the declared cursor.

The Extractor READ function returns the number of rows actually read and an End-of-Cursor Indicator.

This Read function is systematically executed when a cursor OPEN statement is issued.

It may also be executed when a FETCH statement is issued. The purpose of a FETCH is mainly to read the temporary Workfile in order to retrieve a new row.

When the retrieved row is the last one to be read by the Extractor READ function, the function is executed once again if end-of-cursor has not been detected.

ACCESS OPTIMIZATION VIA THE SIZE PARAMETER

The SIZE parameter is a critical factor in the PAF Extractor's efficiency in terms of response time.

SETTING THE SIZE PARAMETER FOR AN ONLINE PAF USER PROGRAM:

The value of the SIZE parameter should not be less than the number of rows fetched per screen (number of records displayed on the screen, for example). Based on a simple hypothesis that all rows which are read are also valid, the optimal value of the SIZE parameter is a multiple of the number of rows fetched per screen.

NOTE: If the online PAF user program includes screen branching operations, the optimal value of the SIZE parameter is equal to the number of rows fetched per screen.

SETTING THE SIZE PARAMETER FOR A BATCH PAF USER PROGRAM:

At first glance, it may seem appropriate to set the SIZE parameter to a large value in order to minimize the number of READs. However, a value which is too large would result in the dynamic creation of too many records for most indexed sequential file access methods.

The ideal in Batch is to sufficiently define the size of the Input/Output BUFFER for the temporary Work File so that the READ function causes only logical input-outputs.

The 'IDENT' Parameter

The purpose of the IDENT parameter is to uniquely identify a conversation in a multi-user environment.

This identification is therefore closely linked to the TP Monitor under which the translated PAF user application will be executed.

The recommendations below take into account the standard variables supplied with the TP Monitor (terminal identifier, etc.).

OS390-CICS or DOS

The EIBTRMID CICS variable identifies each terminal.

IMS/VS

The name of the logical terminal is found in the IO/PCB field (S-IPCB-XNMTE variable in programs generated by the VA Pac OLSD function).

PAF Implementation under VisualAge Pacbase

The DECLARE CURSOR clause must be entered in the WORKING-STORAGE SECTION.

It must therefore be inserted on Work Area (-W) lines in the Program or online Screen.

'EXEC PAF' must start in the 5th position of the LEVEL OR SECTION field, and 'END-EXEC' must be entered after the cursor declaration.

```
A LIN T LEVEL OR SECTION WORK AREA DESCRIPTION
  100       EXEC PAF DECLARE CU01 CURSOR FOR
  120       SELECT * FROM DELDEF
  140       WHERE FDELIL = 5 END-EXEC
```

Except for the DECLARE CURSOR clause, all SQL-PAF statements are written on the Procedural Code (-P) lines of the PAF user Program or online Screen. An EXP PAF Operator generates EXEC PAF and END-EXEC calls, which are found before and after all SQL-PAF statements, respectively.

```
A SF LIN OPE OPERANDS
         EXP OPEN CU01
```

will generate:

```
EXEC PAF
OPEN CU01
END-EXEC
```

The Translated User Program

Before COBOL compilation, the PAF Translator transforms SQL-PAF queries into COBOL declarations and instructions. (Warning: PAF translator is incompatible with a COBOL formatting request).

The EXEC PAF .... END-EXEC sequences are commented out in the COBOL program.

WORKING-STORAGE SECTION

The following phrase:

```
DECLARE <cursor-code> CURSOR FOR SELECT * FROM <table-code>
```

generates the following data declarations in the WORKING- STORAGE SECTION, under Level 01 <cursor-code>-CURSOR, of the PAF user program:

- The Cursor Management field,
- The Specifications field, where the query is translated,
- The Communication Area, i.e., the selected PAF Table.

The two fields which are accessible in the program (Cursor Management Field and Communication Area) are prefixed by the cursor-code. The Specifications field is generated as a FILLER.

EXAMPLE:

Extraction of Text Descriptions (TXTDSC Table) for Text Entity 'TEXT01', including Text Paragraphs greater than 'EE'. The cursor-code is TX04.

```
EXEC PAF  DECLARE TX04 CURSOR FOR SELECT * FROM
          TXTDSC WHERE CTXT = 'TEXT01' AND CPAR > 'EE'
END-EXEC
```

The following fields are grouped under the level:

```
01    TX04-CURSOR
```

CURSOR MANAGEMENT FIELD

```
05      TX04-SAVE.
  10    FILLER        PIC X(06) VALUE 'TX0401'.
  10    TX04-TABCOD   PIC X(10) VALUE 'TXTDSC    '.
  10    TX04-RETCOD   PIC 9(00002).
  10    TX04-ORDER    PIC X(00001).
  10    TX04-FI       PIC X(00001).
```

```
10    TX04-FT       PIC X(00001).
10    TX04-CUSRCU   PIC X(00008).
10    TX04-CPSWCU   PIC X(00008).
10    TX04-CLIBCU   PIC X(00003).
10    TX04-CSESCU   PIC X(00005).
10    TX04-CNETCU   PIC X(00001).
10    TX04-NRECCU   PIC 9(00006).
10    TX04-INTERN   PIC X(00034).
```

where

TABCOD: Table code.

RETCOD: Extractor Return Code.

″0″: No error detected. Other values are presented in chapter 'Error Messages', subchapter 'The PAF Extractor'.

ORDER: Each PAF statement is identified by a number:

1 INIT

2 CONNECT

4 OPEN

6 FETCH

8 CLOSE

9 QUIT

FI: End of cursor READ:

1 Read of cursor's last row,

0 Otherwise.

FT: End of cursor processing:

1 Cursor FETCH beyond its last row,

0 Otherwise.

NOTE: FI is set to '1' in two cases:

- The FETCH command returns the last corresponding record to the cursor selection.

- No data is selected by the OPEN command.

  If a FETCH command is executed when FI = '1', the FT indicator is returned to '1' without the

  new record being used.

  The OPEN command can also change the FI indicator to avoid a useless FETCH.

The other fields contain user parameters related to the CONNECT statement. The values of these parameters are automatically generated in the PROCEDURE DIVISION by the PAF Translator when the CONNECT statement is encountered.

CUSRCU: VisualAge Pacbase user code

CPSWCU: VisualAge Pacbase user password

CLIBCU: VisualAge Pacbase library code

CSESCU: VA PAC database session number (and version)

CNETCU: VisualAge Pacbase database subnetwork option

NRECCU: Max. numb. of records in the Intermed. workfile

INTERN: Internal usage field

NOTE: The Cursor Management field is called <cursor-code>-SAVE since, for an online PAF user program, this field has to be saved when the calling program returns control to the monitor. The Online PAF Extractor actually backs up a representation of the query in the Specifications field <cursor-code>-TECH (see next paragraph) in the Temporary Work File.

For an online PAF user program, the management field is the same as in batch but with two preceding fields, which are used to identify the database and the terminal (see description of the CONNECT statement):

```
10    TX04-IDENT      PIC X(25).
10    TX04-BASE       PIC X(4).
```

The contents of both fields, <cursor-code>-SAVE and <cursor-code>-TECH, must not be modified.

SPECIFICATIONS FIELD

This field is generated at the following level:

```
05      TX04-TECH.
```

This field is a variable length FILLER, where the query is translated for the PAF Extractor. The PAF user cannot access this field.

COMMUNICATION AREA

```
05      TX04.
  10    TX04-LCTX     PIC X(00003).
  10    TX04-SCTX     PIC 9(00004).
  10    TX04-TCTX     PIC X(00001).
  10    TX04-NCTX     PIC X(00001).
  10    TX04-CTXT     PIC X(00006).
  10    TX04-CPAR     PIC X(00002).
  10    TX04-CLIN     PIC 9(00003).
  10    TX04-TLIN     PIC X(00001).
  10    TX04-DLINTX   PIC X(00060).
  10    TX04-CDEL     PIC X(00006).
```

Whatever Table is selected, the first four columns are always generated. They specifiy the origin of the extracted line, i.e.:

LCTX: Library code where the extracted line is defined.

SCTX: Session Number when the line was last modified.

TCTX: Version of session when the line was last modified.

NCTX: Line Source Indicator:

=: The line is extracted from the library to which the cursor is connected.

>: The line is extracted from a higher-level library than the cursor connection library.

<: The line is extracted from a lower-level library than the cursor connection library.

The codes of these four columns may be changed by using the SET statement with the 'CTXCOL' parameter.

The other columns are specific to the selected table.

PROCEDURE DIVISION

For each SQL-PAF statement in the PROCEDURE DIVISION, the following operations occur:

- the TX04-ORDER field is filled in,

- the Extractor subprogram is called and the entire TX04-CURSOR field is passed to it.

When a CONNECT statement is encountered, the Translator sends the user's parameters to the Cursor Management field.

When an OPEN statement is encountered, the Translator fills in (in some cases under specific conditions) the Specifications field (-TECH) when the cursor depends on one or more COBOL fields.

Embedded PAF Cursors

LIMITATIONS OF THE SQL-PAF SYNTAX

The SQL-PAF syntax uses a sub-set of the SQL language. In particular, cursors cannot be defined with embedded SELECT clauses. Embedded SELECT clauses are useful in obtaining information conditioned by a sequence of cross-references such as, a List of Data Elements called in Segments used in Programs.

However, this limitation is not too restrictive since you can declare several cursors (maximum number = 100). When a cursor depends on a COBOL field which belongs to the Communication Area of another cursor, both cursors act as one cursor using embedded SELECT clauses.

EMBEDDED CURSORS: EXAMPLE

Suppose a user wants to obtain, for each Data Structure in the VA Pac Database, the list of Programs that use it.

This query involves the following Tables:

DSTDEF Data Structure Definition,

PGMDST Program Call of Data Structures,

and the following Columns:

```
CDST Data Structure code,
CPGM Program code.
```

The SELECT clause of a standard SQL query is written as follows:

SELECT * FROM PGMDST WHERE

CDST = (SELECT CDST FROM DSTDEF)

With the SQL-PAF syntax, the same query uses two embedded cursors:

DECLARE LCD CURSOR FOR SELECT * FROM DSTDEF

DECLARE PGCD CURSOR FOR SELECT * FROM PGMDST

WHERE CDST = LCD-CDST

The first cursor, coded LCD, provides the list of all Data Structures. The second cursor, coded PGCD, provides the list of Programs using the Data Structure coded LCD-CDST, i.e., the code of the Data Structure currently fetched in the first cursor.

In the PROCEDURE DIVISION, after both cursors are connected, the LCD cursor is opened. As long as LCD-FT is not equal to one (i.e., the LCD cursor is still open), each time a FETCH statement is issued on the LCD cursor, an OPEN statement will be issued on the PGCD cursor. The PGCD cursor is issued. This OPEN allows the code of the next Data Structure to be moved to the LCD-CDST field.

After all FETCH statements are issued for the PGCD cursor, the cursor is closed, and another FETCH statement is issued for the LCD cursor.

In this way, you can simulate, through embedded cursor processing, a single cursor defined by embedded SELECT clauses.

REMINDER:: Up to 100 cursors may be used by a PAF user program.

Execution of PAF User Programs

BATCH

Each time a Batch PAF user program is executed, you have to create and declare the temporary Workfile at the beginning of the job stream (and possibly delete the same file if it was previously created).

This file is an indexed sequential file and has the following characteristics:

- Key length = 12.
- Maximum record size = 1031
- Average record size = 170.

Then, in order to execute the batch PAF user program, you must declare in the JCL the files required for the execution of the extractor (VisualAge Pacbase files and the temporary Workfile.

For more details on the files used by the extractor, refer to the JCL examples provided in the Operations manuals.

ONLINE

The temporary Workfile is an indexed sequential file and has the following characteristics when used with an online PAF user program:

- Key length = 37, starting in position 2.
- Maximum record size = 1161
- Average record size = 200.

Chapter 3. Examples of Programs Using PAF

Introduction

The purpose of this chapter is to present two examples of programs (batch and online) using PAF. Additionally, it suggests ways to use PAF programs (standard quality control, VA Pac database administration, etc.).

The first program example, 'PAFEX1', is a batch program. It builds a list of all the Properties (Data Elements whose Type = 'P') without relational names. Two cursors must be declared: one for the definition of Data Elements, and one for their descriptions. This batch program is a good example of how to use embedded cursors.

The second program example, 'PAFEX2', is an online program. It builds a list of screens that do not conform to a specific local standard (Data Element presentation, initialization character, screen or element help character). This program uses only one cursor (for the screen definition), and manages screen scrolling. This online program provides an example of how to transmit the PAF context between each iteration of a dialogue.

Batch Example

OBJECTIVE:

- To list the Properties without relational names.

  PAF Cursor Declarations:

  – CU01 selects 'P'-type Data Elements (Properties).

  – CU02 (opened for each Element found by CU01) selects the description lines which contain a relational name.

  Procedural logic:

  – F02BA: PAF initialization.

  – F02CA: CU01 Cursor Connection. The user code and password are hard-coded but could be obtained through a Read of an input file. The sub-network option is set to 'U', which indicates that only Properties in Library 'CIV' are taken into account. The CONNECT statement establishes the context for the PAF Read.

  – FO2DA: CU02 Cursor Connection.

  – F21BA: Opening the CU01 Cursor. This statement involves reading the 'P'-type Data Element Definition screens in Library 'CIV', and writing them in the PAF work file.

- F21CA: Fetching the definition screens, i.e., the screens are read one-by-one from the PAF work file.
- F21DA: As long as the end-of-cursor is not reached (CU01-FI = '0'), the CU02 Cursor is opened for each fetched element. This Cursor selects only the 'R'-type description lines of the fetched elements. Therefore, an immediate end-of-cursor (CU02-FI = '1') after the first FETCH means that this property does not have a relational name. In this case, a line is formatted and printed on a Report. The CU02 Cursor is closed so that it can be reopened for the next element of the CU01 Cursor.
- F79 : When all of the Properties have been FETCHED, the CU01 Cursor is closed and a QUIT statement is issued in order to close the database files and the PAF work file.

```
                  APPLI  CICS/VSAM                        *PTJML.D474.CIV.2020
PROGRAM DEFINITION....... PAFEX1

PROGRAM NAME.....................: LIST PROPERTIES W/O SQL NAME

CODE FOR SEQUENCE OF GENERATION....: PAFEX1

TYPE OF CODE TO GENERATE...........: 0
COBOL NUMBERING AND ALIGNMENT OPT..:
CONTROL CARDS IN FRONT OF PROGRAM..:
CONTROL CARDS IN BACK OF PROGRAM...:
COBOL PROGRAM-ID...................: PAFEX1
MODE OF PROGRAMMING................: P
TYPE AND STRUCTURE OF PROGRAM......: B
PROGRAM CLASSIFICATION CODE........: P   PROGRAM
TYPE OF PRESENCE VALIDATION........:
SQL INDICATORS GENERATION WITH '-'.:


EXPLICIT KEYWORDS..:
UPDATED BY.........:            ON :            AT:   :  :      LIB :
SESSION NUMBER.....: 2013       LIBRARY......: CIV    LOCK....:

O: C1 CH: Ppafex1                    ACTION:
```

```
                    APPLI  CICS/VSAM                        *PTJML.D474.CIV.2020
DATA STRUCTURES USED IN PROGRAM :    PAFEX1 LIST PROPERTIES W/O SQL NAME

A DP CO : DL EXTERN OARFU BLOCK T  B M U RE SE L UNIT C  SELECTION F E R L PL
  PR    : PR PRLIST SSFOU     0 R      I                 A         I     1
        :    STAT.FLD:             ACC. KEY:             RECTYPEL
        :
        :    STAT.FLD:             ACC. KEY:             RECTYPEL
        :
        :    STAT.FLD:             ACC. KEY:             RECTYPEL
        :
        :    STAT.FLD:             ACC. KEY:             RECTYPEL
        :
        :    STAT.FLD:             ACC. KEY:             RECTYPEL
        :
        :    STAT.FLD:             ACC. KEY:             RECTYPEL
        :
        :    STAT.FLD:             ACC. KEY:             RECTYPEL
        :
        :    STAT.FLD:             ACC. KEY:             RECTYPEL
        :
        :    STAT.FLD:             ACC. KEY:             RECTYPEL

O: C1 CH: -CD
```

```
                  APPLI  CICS/VSAM                        *PTJML.D474.CIV.2020
WORK AREAS.....ENTITY TYPE P PAFEX1 LIST PROPERTIES W/O SQL NAME

CODE FOR PLACEMENT..:      BA
A LIN T LEVEL OR SECTION WORK AREA DESCRIPTION                          OCCURS
  100 * PROPERTIES LIST
  110       EXEC PAF DECLARE CU01 CURSOR FOR
  120       SELECT * FROM DELDEF WHERE
  130       TDEL = 'P'
  140       END-EXEC
  200 * LIST OF RELATIONAL NAMES OF A PROPERTY
  210       EXEC PAF DECLARE CU02 CURSOR FOR
  220       SELECT * FROM DELDSC WHERE
  230       CDEL = CU01-CDEL AND
  240       TLIN = 'R'
  250       END-EXEC









O: C1 CH: -W
```

```
                    APPLI  CICS/VSAM                        *PTJML.D474.CIV.2020
PROCEDURAL CODE  P PAFEX1 LIST PROPERTIES W/O SQL NAME        FUNCTION: 02

A SF LIN OPE OPERANDS                          LVTY CONDITION
         N   INITIALIZATION AND CONNECTIONS    05BL
- -- --- --- -------------------------------- ---- --------------------------
  BA     N   INITIALIZATION                    10BL
  BA 100 EXP INIT
- -- --- --- -------------------------------- ---- --------------------------
  CA     N   CONNECTION OF CU01                10BL
  CA 100 EXP CONNECT CU01 TO
  CA 110     USER    = 'USER'
  CA 120     PASS    = 'PASS'
  CA 130     LIB     = 'CIV'
  CA 140     NET     = 'U'
  CA 150     SESSION = SPACES
  CA 160     SIZE    = 40
- -- --- --- -------------------------------- ---- --------------------------
  DA     N   CONNECTION OF CU02                10BL
  DA 100 EXP CONNECT CU02 TO
  DA 110     USER    = 'USER'
  DA 120     PASS    = 'PASS'

O: C1 CH: -P02
```

```
                  APPLI  CICS/VSAM                        *PTJML.D474.CIV.2020
PROCEDURAL CODE  P PAFEX1 LIST PROPERTIES W/O SQL NAME        FUNCTION: 02

A SF LIN OPE OPERANDS                          LVTY CONDITION
  DA 130     LIB     = 'CIV'
  DA 140     NET     = 'U'
  DA 150     SESSION = SPACES
  DA 160     SIZE    = 1
- -- --- --- ------------------------------- ---- -------------------------
```

```
O: C1 CH: -P02da
```

```
                  APPLI  CICS/VSAM                        *PTJML.D474.CIV.2020
PROCEDURAL CODE  P PAFEX1 LIST PROPERTIES W/O SQL NAME        FUNCTION: 21

A SF LIN OPE OPERANDS                          LVTY CONDITION
         N   PROCESS                           05BL
- -- --- --- ------------------------------- ---- --------------------------
  BA     N   OPEN THE PROPERTY LIST           10BL
  BA 100 EXP OPEN CU01
- -- --- --- ------------------------------- ---- --------------------------
  CA     N   READ EACH PROPERTY               10DW CU01-FI = '0'
  CA 100 EXP FETCH CU01
- -- --- --- ------------------------------- ---- --------------------------
  DA     N   SEARCH FOR RELATIONAL NAMES      15BL
  DA 100 *   OPEN RELATIONAL NAMES
  DA 110 EXP OPEN CU02
  DA 200 *   READ RELATIONAL NAMES
  DA 210 EXP FETCH CU02
  DA 300 *   NO RELATIONAL NAME : REPORT      99IT CU02-FT = '1'
  DA 310 P   F8F
  DA 400 *   CLOSE RELATIONAL NAMES           99BL
  DA 410 EXP CLOSE CU02
- -- --- --- ------------------------------- ---- --------------------------

O: C1 CH: -P21
```

```
                  APPLI  CICS/VSAM                        *PTJML.D474.CIV.2020
PROCEDURAL CODE  P PAFEX1 LIST PROPERTIES W/O SQL NAME        FUNCTION: 79

A SF LIN OPE OPERANDS                          LVTY CONDITION
         N   DISCONNECT FROM PAF               05BL
     100 EXP CLOSE CU01
     110 EXP QUIT
     120 GFT
- -- --- --- ------------------------------- ---- -------------------------









*** END ***
O: C1 CH: -P79
```

```
                  APPLI  CICS/VSAM                        *PTJML.D474.CIV.2020
REPORT CODE             PRA

NAME..................: LIST PROPERTIES W/O SQL NAME

COMMENTS..............:

NATURE................: E REPORT
PRINTER TYPE..........: L

LINE LENGTH...........: 132
FORMAT FOR TOTALS     : INTEGER........: 11
                        DECIMAL PLACES.: 07




EXPLICIT KEYWORDS..:


UPDATED BY.........:             ON :            AT:   :  :      LIB :
SESSION NUMBER.....: 2013        LIBRARY......: CIV    LOCK....:

O: C1 CH: Rpra                        ACTION:
```

```
                  APPLI  CICS/VSAM                        *PTJML.D474.CIV.2020
REPORT LAYOUT :          PRA LIST PROPERTIES W/O SQL NAME          LENGTH= 132

A LN CP S           1    1    2    2    3    3    4    4    5    5    6    6
           1...5....0....5....0....5....0....5....0....5....0....5....0....5.
  00  1
  03  2                          LIST OF PROPERTIES WITHOUT RELATIONAL NAME
  06  3                          ----------------------------------------
  09  4    ------------------------------------------------------------------
  12  5    I  CODE  I           PROPERTY NAME              I INP.  FORM. I IN
  15  4    ------------------------------------------------------------------
  18  6    I        I                                     I             I
  21  4    ------------------------------------------------------------------

O: C1 CH: -L
```

```
                  APPLI  CICS/VSAM                        *PTJML.D474.CIV.2020
REPORT LAYOUT :          PRA LIST PROPERTIES W/O SQL NAME          LENGTH= 132

A LN CP S  6    7    7    8    8    9    9   10   10   11   11   12   12   13
           5....0....5....0....5....0....5....0....5....0....5....0....5....0
  03  2    E                                        PAGE:
  06  3    -
  09  4    ------------------------------------------------------------------
  12  5    INT. FORM. I U I        OUTPUT FORMAT        I PARENT I LIB I SESS
  15  4    ------------------------------------------------------------------
  18  6              I   I                              I        I     I
  21  4    ------------------------------------------------------------------
```

```
                  APPLI  CICS/VSAM                        *PTJML.D474.CIV.2020
REPORT DESCRIPTION:       PRA LIST PROPERTIES W/O SQL NAME

A:   LINE LENGTH:    132 LI PAGE:   60 CAT TBL INST:    WR OPT:   SECTION: 00
     COMMENTS....:               CONDITIONS

A CA LIN T TLI ST CP SKP FUSF COMMENTS      CONDITIONS
  BA 010        1 01 01*      HEADER        5-PR00-ALC NOT < 5-PR00-ALCM
  BA 030          02 01
  BA 050          03 01
  BA 070          04 01
  BA 090          05 01
  BA 110          04 01
- -- --- - --- -- -- --- ---- ------------- ---------------------------------
  CA 010        2 06 01       DETAIL
- -- --- - --- -- -- --- ---- ------------- ---------------------------------
  DA 010          04 01       FOOTER        5-PR00-ALC NOT < 5-PR00-ALCM OR
  DA 020                                    CU01-FI = '1'
- -- --- - --- -- -- --- ---- ------------- ---------------------------------




O: C1 CH: -D
```

```
                    APPLI  CICS/VSAM                        *PTJML.D474.CIV.2020
REPORT CALL OF ELEMENTS  PRA LIST PROPERTIES W/O SQL NAME

A ST ELEM   L : STA C O W SOURCE    FLD CONDITION                        LIBR.
  01 XPAGE  0 : 112   M 5 PR00-APC                                       2013
- -- ------ - : --- - - - ------------ ------------------------------ -----
  02 CDEL   0 :   3   M   CU01CDEL                                      2013
  02 LDEL   0 :  12   M   CU01LDEL                                      2013
  02 FDELT  0 :  51   M   CU01FDELT                                     2013
  02 FDELI  0 :  65   M   CU01FDELI                                     2013
  02 ODELUS 0 :  78   M   CU01ODELUS                                    2013
  02 FDELO  0 :  82   M   CU01FDELO                                     2013
  02 CDELP  0 : 112   M   CU01CDELP                                     2013
  02 LCTX   0 : 121   M   CU01LCTX                                      2013
  02 SCTX   0 : 127   M   CU01SCTX                                      2013
  02 TCTX   0 : 131   M   CU01TCTX                                      2013
- -- ------ - : --- - - - ------------ ------------------------------ -----
               :
               :
               :
               :
               :
*** END ***
O: C1 CH: -CE
```

Online Example

OBJECTIVE:

- To list the Screens that do not conform to a particular local standard.

  PAF Cursor Declarations:

  – CU01 selects those Screen Definitions in which the Data Element presentation, initialization character, or Screen or Element help character differs from the standard.

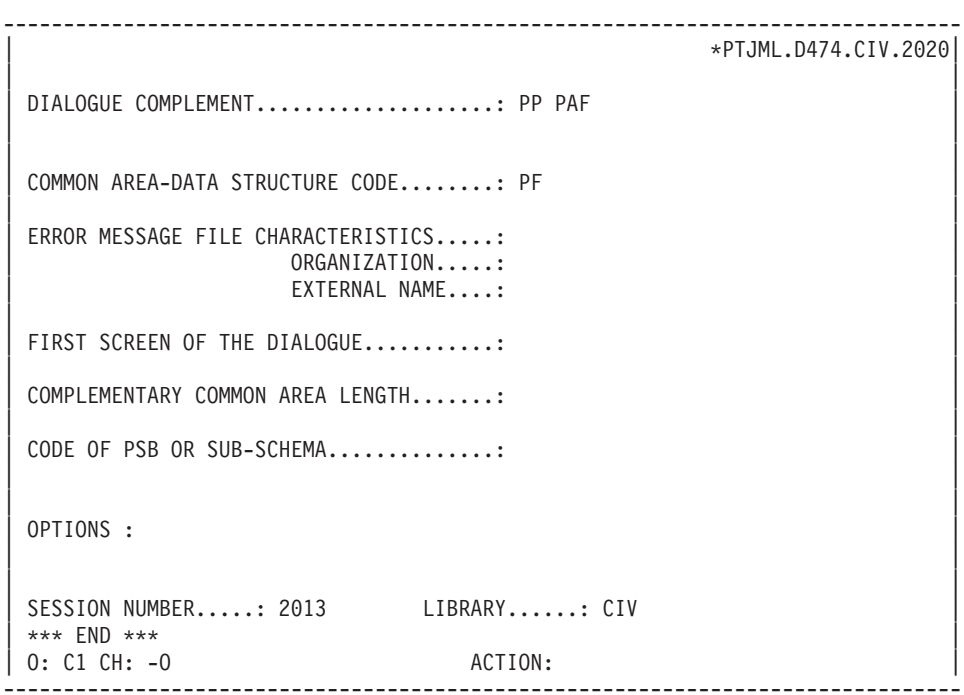
  Procedural logic:

  – F02 : PAF initialization only for the first entry into the program (EIBCALEN = '0').

  – F06CA: CU01 Cursor Connection. The user code and password are hard-coded but could be entered on a menu and passed via the COMMAREA. The VA Pac Database code must be specified (D474). In addition, the terminal code (EIBTRMID) is assigned to the PAF 'IDENT' parameter in order to ensure that the keys between conversations in the PAF file are unique. The 'SIZE' parameter is set to '10', which corresponds to the number of repeated lines on the screen. Only the number of screens necessary for display are read, in order to avoid online reads that are too long and may prove to be pointless if a user ends up exiting the transaction without consulting the whole list.

In this example, the F06 function is executed only on the first entry into the program. Thus, the CONNECT and OPEN statements are issued only once. This example does not take into account interactive modifications of selection criteria. This could be implemented by modifying specific screen-top category fields that are associated with the criterion of the SELECT statement defining the CU01 cursor.

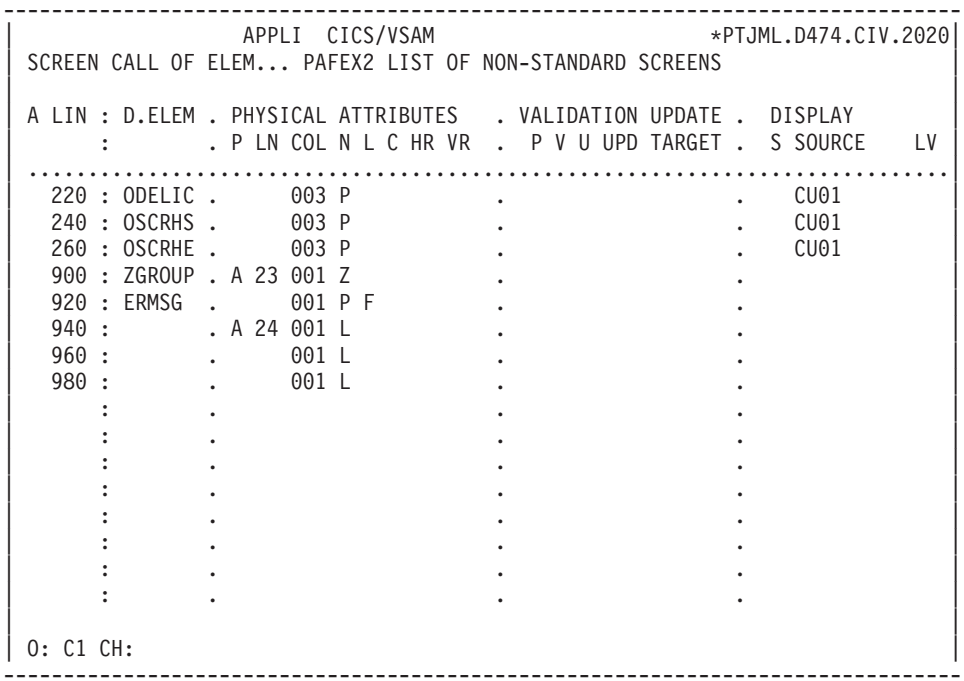
- F06DA: Opening the CU01 Cursor. This statement causes screen definitions which do not conform to a local standard to be read and stored in the PAF work file.
- F06EA: The field to save the CU01 Cursor (CU01-SAVE) is transferred to a backup field in the COMMAREA.
- F4031: Closing the CU01 Cursor and executing the PAF CLOSE and QUIT statements.
- F52BA: Retrieving the CU01-SAVE field from the COMMAREA. If all records have been fetched, the Cursor is closed.
- F60PF: If the last record has not yet been fetched, and as long as the repetitive category of the screen is being processed, the next record is fetched.
- F60PQ: If the last record has been fetched, the 'END' message is written and an exit from the iteration is performed.
- F75 : If no error is detected, the field to save the CU01 Cursor (CU01-SAVE) is transferred to a backup field in the COMMAREA.


*PTJML.D474.CIV.2020
DIALOGUE COMPLEMENT.....: PP PAF
COMMON AREA-DATA STRUCTURE CODE.....: PF
ERROR MESSAGE FILE CHARACTERISTICS.....:
ORGANIZATION.....:
EXTERNAL NAME....:
FIRST SCREEN OF THE DIALOGUE.....:
COMPLEMENTARY COMMON AREA LENGTH.....:
CODE OF PSB OR SUB-SCHEMA.....:
OPTIONS :
SESSION NUMBER.....: 2013
LIBRARY.....: CIV
*** END ***
O: C1 CH: -O
ACTION:

```
|                   APPLI  CICS/VSAM                        *PTJML.D474.CIV.2020|
| ON-LINE SCREEN DEFINITION.....: PAFEX2                                        |
|                                                                               |
| SCREEN NAME.....................: LIST OF NON-STANDARD SCREENS                |
|                                                                               |
| SCREEN SIZE (LINES, COLUMNS) ......:   24       080                          |
| LABEL TYPE, TABS, INITIALIZATION...: * S      * 02                            |
| HELP CHARACTER SCREEN, DATA ELEMENT: * 11     * 12                            |
|                                                                               |
|                                    LABELS   DISPLAY INPUT ER.MESS. ER.FLD.    |
| INTENSITY ATTRIBUTE ..............:   N        N       N       B       B      |
| PRESENTATION ATTRIBUTE ...........:   N        N       N       N       N      |
| COLOR ATTRIBUTE ..................:   W        W       W       W       W      |
|                                                                               |
| TYPE OF COBOL AND MAP TO GENERATE..:   0 * 0   IBM OS CICS (PROG. & MAP BMS)  |
| CONTROL CARD OPTIONS FRONT & BACK..: * CC       (PROGRAM)   * KK       (MAP) |
| EXTERNAL NAMES ...................:   PF001P   (PROGRAM)     PF001M   (MAP) |
| TRANSACTION CODE..................:                                           |
| EXPLICIT KEYWORDS..:                                                          |
| UPDATED BY.........:            ON :            AT:   :  :      LIB :         |
| SESSION NUMBER.....: 2013       LIBRARY......: CIV    LOCK....:               |
|                                                                               |
| O: C1 CH: Opf0010                    ACTION:                                  |
```

```
------------------------------------------------------------------------------
|                   APPLI  CICS/VSAM                      *PTJML.D474.CIV.2020|
| ON-LINE SCREEN X-REF'S TO SCREENS FOR ON-LINE SCREEN : PAFEX2               |
|                                                                             |
| SCREEN : LIN  D.ELEM  P LN COL N P C HR VR  C P O SEG D.ELEM  W SEG D.ELEM LV|
| ----------------------------------------------------------------------------|
| ------- CALL OF  T  TYPE  (TITLE) -----------------------------------------|
| PAFEX2   020  PAFEX2  A 01 025 T     00 00                                  |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
| O: C1 CH: -XO                                                               |
------------------------------------------------------------------------------
```

```
                  APPLI  CICS/VSAM                        *PTJML.D474.CIV.2020
SCREEN CALL OF ELEM... PAFEX2 LIST OF NON-STANDARD SCREENS

A LIN :  D.ELEM  . PHYSICAL ATTRIBUTES   . VALIDATION UPDATE .  DISPLAY
      :          . P LN COL N L C HR VR  .  P V U UPD TARGET .  S SOURCE    LV
.....................................................................................
  010 : PFKEY   .          V             .    O V A          .    EN
  011 :         .                        .      V S          .    01
  012 :         .                        .      G            .    02
  020 : PAFEX2  . A 01 025 T             .                   .
  025 : PFVIEW  .   02 025 V             .  R                .
  040 :         .   02 015 L             .                   .
  045 :         .      001 L             .                   .
  060 : ODELL1  .   02 001 V             .  R                .
  080 : ODELI1  .          V             .  P                .
  100 : OSCRH1  .   01 001 V             .  P                .
  120 : OSCRH2  .          V             .  P                .
  130 :         .   01 001 L             .                   .
  140 : RGROUP  .   01 001 R 3      10    .                   .
  160 : CSCR    .      003 P             .                   .    CU01
  180 : LSCR    .      003 P             .                   .    CU01
  200 : ODELLB  .      003 P             .                   .    CU01

O: C1 CH: -CE
```


APPLI CICS/VSAM *PTJML.D474.CIV.2020
SCREEN CALL OF ELEM... PAFEX2 LIST OF NON-STANDARD SCREENS
A LIN : D.ELEM . PHYSICAL ATTRIBUTES . VALIDATION UPDATE . DISPLAY
: . P LN COL N L C HR VR . P V U UPD TARGET . S SOURCE LV
220 : ODELIC . 003 P . . CU01
240 : OSCRHS . 003 P . . CU01
260 : OSCRHE . 003 P . . CU01
900 : ZGROUP . A 23 001 Z . .
920 : ERMSG . 001 P F . .
940 : . A 24 001 L . .
960 : . 001 L . .
980 : . 001 L . .
O: C1 CH:

```
                  APPLI  CICS/VSAM                        *PTJML.D474.CIV.2020
SCREEN CALL OF ELEM... PAFEX2 LIST OF NON-STANDARD SCREENS

A LIN : D.ELEM . PHYSICAL ATTRIBUTES         . LABEL
      :        . P LN COL N L HR VR IN PR CO . T LITERALS
  010 : PFKEY  .          V                  .
  011 :        .                             .
  012 :        .                             .
  020 : PAFEX2 . A 01 025 T                  .
  025 : PFVIEW .   02 025 V                  . I C
  040 :        .   02 015 L                  .   PLEASE ENTER THE STANDARD/
  045 :        .      001 L                  .   SCREEN CHARACTERISTICS:/
  060 : ODELL1 .   02 001 V                  . I S
  080 : ODELI1 .          V                  . I _
  100 : OSCRH1 .   01 001 V                  . I 11
  120 : OSCRH2 .          V                  . I 12
  130 :        .   01 001 L                  . A 079-
  140 : RGROUP .   01 001 R 3    10          .
  160 : CSCR   .      003 P                  .
  180 : LSCR   .      003 P                  .
  200 : ODELLB .      003 P                  .

O: C2 CH: -CE
```

```
                  APPLI  CICS/VSAM                        *PTJML.D474.CIV.2020
SCREEN CALL OF ELEM... PAFEX2 LIST OF NON-STANDARD SCREENS

A LIN : D.ELEM . PHYSICAL ATTRIBUTES         . LABEL
      :        . P LN COL N L HR VR IN PR CO . T LITERALS
.....................................................................
  220 : ODELIC .      003 P                  .
  240 : OSCRHS .      003 P                  .
  260 : OSCRHE .      003 P                  .
  900 : ZGROUP . A 23 001 Z                  .
  920 : ERMSG  .      001 P F                .
  940 :        . A 24 001 L                  .   ENTER: DISPLAY, PF1: SCRLL,/
  960 :        .      001 L                  .   PF2: EXIT, PF11: SCRN HELP,/
  980 :        .      001 L                  .   PF12: FIELD HELP/
      :        .                             .
      :        .                             .
      :        .                             .
      :        .                             .
      :        .                             .
      :        .                             .
      :        .                             .
      :        .                             .

O: C2 CH: -CE
```

```
                    APPLI  CICS/VSAM                    *PTJML.D474.CIV.2020
                                                                 S
SEGMENT CALL OF ELEMENTS PF00 COMMUNICATION AREA

A LIN : ELEM.  INT.FORM. U OCC GR K CMD456 CONT VALUE/SFC UPD/TRGET DOC LIBR
  100 : PFSAVE X(115)    D
  110 : PFFRST X         D
      :
      :
      :
      :
      :
      :
      :
      :
      :
      :


O: C1 CH: Spf00CE
```

```
                  APPLI  CICS/VSAM                      *PTJML.D474.CIV.2020
WORK AREAS.....ENTITY TYPE O PAFEX2 LIST OF NON-STANDARD SCREENS

CODE FOR PLACEMENT..:      BA
A LIN T LEVEL OR SECTION WORK AREA DESCRIPTION                          OCCURS
  100 * LIST OF NON STANDARD SCREENS
  110       EXEC PAF DECLARE CU01 CURSOR FOR
  120       SELECT * FROM SCRDEF WHERE
  130       ODELLB <> I-0020-ODELL1 OR
  140       ODELIC <> I-0020-ODELI1 OR
  150       OSCRHS <> I-0020-OSCRH1 OR
  160       OSCRHE <> I-0020-OSCRH2 OR
  170       END-EXEC









O: C1 CH: -W
```

```
                  APPLI  CICS/VSAM                      *PTJML.D474.CIV.2020
PROCEDURAL CODE       O PAFEX2 LIST OF NON-STANDARD SCREENS   FUNCTION: 02

A SF LIN OPE OPERANDS                          LVTY CONDITION
         N   PAF INITIALIZATIONS               05IT EIBCALEN = 0
     100 EXP INIT
     110 M   '0'     PF00-PFFRST
- -- --- --- -------------------------------- ---- -------------------------
O: C1 CH: -P02
```

```
------------------------------------------------------------------------------
|                   APPLI  CICS/VSAM                      *PTJML.D474.CIV.2020|
| PROCEDURAL CODE       O PAFEX2 LIST OF NON-STANDARD SCREENS   FUNCTION: 06   |
|                                                                              |
| A SF LIN OPE OPERANDS                          LVTY CONDITION                |
|          N   PAF CONNECTION                    05IT PF00-PFFRST = '0'        |
| - -- --- --- ------------------------------- ---- --------------------------|
|   BA     N   SET FIRST TIME INDICATOR         10BL                          |
|   BA 100 M   '1'   PF00-PFFRST                                              |
| - -- --- --- ------------------------------- ---- --------------------------|
|   CA     N   CURSOR CONNECTION                10BL                          |
|   CA 100 EXP CONNECT CU01 TO                                                |
|   CA 110     USER    = 'USER'                                               |
|   CA 120     PASS    = 'PASS'                                               |
|   CA 130     LIB     = 'CIV'                                                |
|   CA 140     SESSION = SPACES                                               |
|   CA 150     NET     = I-0010-PFVIEW                                        |
|   CA 160     SIZE    = 10                                                   |
|   CA 170     BASE    = 'D474'                                               |
|   CA 180     IDENT   = EIBTRMID                                             |
| - -- --- --- ------------------------------- ---- --------------------------|
|   DA     N   OPEN CURSOR                      10BL                          |
|   DA 100 EXP OPEN CU01                                                      |
|                                                                             |
| O: C1 CH: -P06                                                              |
|                                                                             |
------------------------------------------------------------------------------
```

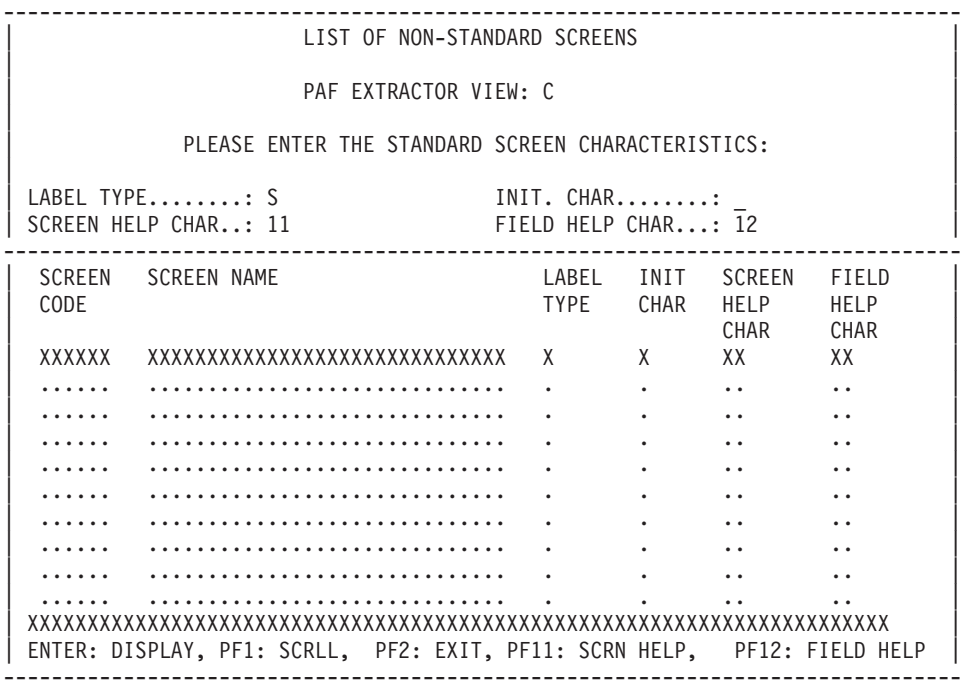
```
                    APPLI  CICS/VSAM                        PTJML.D474.CIV.2020
 PROCEDURAL CODE       O PAFEX2 LIST OF NON-STANDARD SCREENS   FUNCTION: 06

 A SF LIN OPE OPERANDS                          LVTY CONDITION
   EA     N   SAVE CURSOR IN COMMAREA           10BL
   EA 100 M   CU01-SAVE PF00-PFSAVE
 - -- --- --- ------------------------------- ---- -------------------------
 O: C1 CH: -P06ea
```

```
                  APPLI  CICS/VSAM                        *PTJML.D474.CIV.2020
PROCEDURAL CODE       O PAFEX2 LIST OF NON-STANDARD SCREENS   FUNCTION: 40

A SF LIN OPE OPERANDS                          LVTY CONDITION
  31     N   END OF PAF PROCESSING             15BL
  31 100 EXP CLOSE CU01
  31 200 EXP QUIT
- -- --- --- -------------------------------- ---- --------------------------




O: C1 CH: -P40
```

```
                  APPLI  CICS/VSAM                        *PTJML.D474.CIV.2020
PROCEDURAL CODE       O PAFEX2 LIST OF NON-STANDARD SCREENS   FUNCTION: 52

A SF LIN OPE OPERANDS                          LVTY CONDITION
         N   REINITIALIZATION OF PAF AREAS    05BL
     100 M   PF00-PFSAVE CU01-SAVE
- -- --- --- -------------------------------- ---- --------------------------
  BA     N   CLOSE IF END OF CURSOR           10IT CU01-FT = '1'
  BA 100 EXP CLOSE CU01
- -- --- --- -------------------------------- ---- --------------------------









O: C1 CH: -P52
```

```
                  APPLI  CICS/VSAM                        *PTJML.D474.CIV.2020
PROCEDURAL CODE       O PAFEX2 LIST OF NON-STANDARD SCREENS   FUNCTION: 60

A SF LIN OPE OPERANDS                          LVTY CONDITION
  PF     N   FETCH SCREEN DEFINITION RECORDS  10IT CU01-FT = '0'
  PF 100 EXP FETCH CU01                         AN CATX = 'R'
- -- --- --- -------------------------------- ---- --------------------------
  PQ     N   DISPLAY END OF LIST MESSAGE      10IT CU01-FT = '1'
  PQ 100 ERU 0001
  PQ 200 GFT
- -- --- --- -------------------------------- ---- --------------------------









O: C1 CH: -P60
```

```
                  APPLI  CICS/VSAM                        *PTJML.D474.CIV.2020
PROCEDURAL CODE       O PAFEX2 LIST OF NON-STANDARD SCREENS   FUNCTION: 75

A SF LIN OPE OPERANDS                          LVTY CONDITION
         N   SAVE CURSOR IF NO ERROR           05IT GR-EG = '1'
     100 M   CU01-SAVE PF00-PFSAVE               AN CU01-FT = '0'
- -- --- --- -------------------------------- ---- --------------------------
*** END ***
O: C1 CH: -P75
```


LIST OF NON-STANDARD SCREENS
PAF EXTRACTOR VIEW: C
PLEASE ENTER THE STANDARD SCREEN CHARACTERISTICS:
LABEL TYPE.....: S
INIT. CHAR.....: _
SCREEN HELP CHAR..: 11
FIELD HELP CHAR...: 12
SCREEN CODE
SCREEN NAME
LABEL TYPE
INIT CHAR
SCREEN HELP CHAR
FIELD HELP CHAR
XXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
ENTER: DISPLAY, PF1: SCRLL, PF2: EXIT, PF11: SCRN HELP, PF12: FIELD HELP

Chapter 4. PUF - Pacbase Update Facility

Batch Mode - UPDP / UPGP

The Pacbase Update Facility (PUF) procedure enables to update a VA Pac development or administration database with extractions of tables and columns carried out by PAF.

The UPDP and UPGP procedures enable to update a development database or the administration database respectively from a sequential file.

For a description of these procedures and their user input, refer to the following manuals:

- 'The Developer's Procedures' Manual, chapter 'Batch Update', subchapter 'UPDP - Update from PAF Tables'.
- 'The Administrator's Procedures' Manual, chapter 'Administration Database Management' subchapter 'UPGP - Updates with PAF'.

Online Mode

In online mode, PUF enables to perform updates, in real-time, of large transactions in VisualAge Pacbase from a PAF User Program. Problems of concurrent batch and online updates are thus be solved on a number of platforms.

THE SERVICE

The PUF communication program receives and sends, via a middleware, messages related to several applications.

Following an extraction made by PAF, the client application requests the modification of one or several entities and then launches the update. The transactions concerned are then transferred in the SYSPAF file. This flow of transactions can contain several user identification lines (* lines).

Once all the transactions are transferred, the PUF service is called. It performs the updates and if errors are detected, it sends back the erroneous transactions in the SYSPAF file.

The client application requests the following types of services:

- Entity extraction (download): the monitor sends the entity description in the communication area. There are two types of download:

- download with intention of updating: the download description is locked until the upload forbidding all other updating.
- download for consultation: no lock in place, the entity remains modifiable.

- Entity update (upload): the monitor receives an entity description which it transfers in the SYSPAF file when the last message is sent; it starts the uploading transaction and the request to unblock the entity. It then sends the erroneous transactions with errors. This service is stored in the CHOICE field, i.e. the communication field; it is 18 characters long and is coded as follows:
  - UPDOWN (1 char.): D for download, U for upload, V V for lock.
  - SLASH (2 chars.): fixed value //
  - COMET (1 char.): method code
  - DLIST (2 chars.): LC for list by code, LT for list by type, LU for user list (only xxFCOM communication monitor)
  - DSCODE (3 chars.): local entity code
  - ENTITE (6 chars.): entity code
  - DSCHXT (2 chars.): entity description selection ** for the whole of the entity, blank for the definition page
  - UPDWV (1 char.): this variable depends on UPDOWN.
    - in download = W if intending to update and X if not intending to update (lock positioned)
    - in upload = U with unlocking at the end of updating, V without unlocking after updating.
    - in locking = D for unlocking request, V for locking request.
- Locking/unlocking of a technical lock on a number of entity descriptions.

USER INPUT

The F000 suffixed Program is used as a dispatcher for the PUF-TP application. For the transfer of transactions, the dispatcher uses the PA suffixed temporary file which is the PAF workfile; it sends the update transactions to the various programs.

In the workfile, the records used are the following:

- PA70 which contains the transactions sent by the User program to the dispatcher. These are concelled by the dispatcher.
- PA80 which contains the erroneous transactions (with the error label and gravity code), sent by the dispatcher to the User Program. After analysis, this latter takes in charge the cancellation of PA80s.

In case of 'ABEND', the dispatcher sends a message in the communication area to warn the User Program.

The message appears in the following format:

- ABEND
- return code (2 char.):
  31 = problem with the PA file
  32 = problem with the VisualAge Pacbase files
  99 = problem with a program
- external name of the file or the program (8 char.).

THE PAF CLIENT

The procedure for using PUF online is described in chapter 'Implementation in User Programs', subchapter 'Syntax of the SQL-PAF Language'. This procedure follows the following specific statements:

- INSERT: storing of the information

- CALPUF: triggering of the update via the call of the dispatcher

and possibly

- FETCHER: recovering of errors signaled by PUF online in the PA80 file.

THE REMOTE CLIENT (REMOTE PUF)

When PUF acts as a server, it can be used from a remote client: it is the Remote-PUF.

It is possible to use the middleware communications. The user must create his/her own communication monitor whose object will be to receive the message coming from the client and to transmit the information to the dispatcher. This information must be stored in the PAF work file.

The dialog monitor has a communication area whose structure is the same as that of the screens generated by Pacbench C/S. It is activated by the client application.

List of Statements and How They Work

PUF CURSOR-INDEPENDENT STATEMENTS (ONLINE ONLY)

To use PUF, refer to the specific INSERT, CALPUF and FETCHER statements, as well as the UPDATE clause and the SET statement described in chapter 'Implementation in User Programs', subchapter 'Syntax of the SQL-PAF Language'.

On coming across the 'UPDATE = pref' clause, the following fields are generated:

```
01          pref-PUFCOM.
  05        pref-PUFID   PIC X(25)  VALUE SPACE.
  05        pref-PUFRET.
    10      pref-PUFPB   PIC X(05)  VALUE SPACE.
    10      pref-PUFRC   PIC X(02)  VALUE SPACE.
    10      pref-PUFNX   PIC X(08)  VALUE SPACE.
    10      pref-FILLER  PIC X(15)  VALUE SPACE.
  05        pref-PUFTRA  PIC X(04)  VALUE SPACE.
  05        pref-FILLER  PIC X(11)  VALUE SPACE.
```

PUFID: is the identifier of the online conversation using PAF. (similiar to the IDENT parameter of the CONNECT statement)

PUFRET: contains the fields returned by PUF:

PUFPB: a problem was detected (ABEND or ERROR).

PUFRC: return code.

PUFNX: external name of the file responsible.

FILLER: (15 chars. long) - not in use at present.

PUFTRA: is the transaction code of the VA Pac Database (similiar to the BASE parameter of the CONNECT statement)

```
01          pref-PAFCOM.
  05        pref-PAFID   PIC X(25)  VALUE SPACE.
  05        pref-PAFTRA  PIC X(04)  VALUE SPACE.
  05        pref-FILLER  PIC X(16)  VALUE SPACE.
  05        pref-PAFRC   PIC X(02)  VALUE SPACE.
  05        pref-ORDER   PIC X(01)  VALUE SPACE.
  05        pref-FI      PIC X(01)  VALUE SPACE.
  05        pref-FT      PIC X(01)  VALUE SPACE.
  05        pref-PUFERR.
    10      pref-PUFENR.
      15    pref-PUFMVT  PIC X(01)  VALUE SPACE.
      15    pref-PAFTAB  PIC X(10)  VALUE SPACE.
      15    pref-PAFEXT  PIC X(1055) VALUE SPACE.
    10      pref-PUFGRE  PIC X(01)  VALUE SPACE.
    10      pref-PUFLIE  PIC X(66)  VALUE SPACE.
  05        pref-NUMERR  PIC 9(06)  VALUE ZERO.
  05        pref-NUMENR  PIC 9(06)  VALUE ZERO.
  05        pref-FILLER  PIC X(11)  VALUE SPACE.
```

PAFID: is the identifier of the online conversation using PAF (similiar to the IDENT parameter of the CONNECT statement)

PAFTRA: is the transaction code of the VA Pac Database. (similiar to the BASE parameter of the CONNECT statement)

FILLER: (16 chars. long) - unused.

PAFRC: is the return code of the PAF extractor (70 on INSERT: the recording already exists)

ORDER: is the statement code for the PAF extractor ('I' for INSERT; 'E' for FETCHER)

FI: is the end of reading errors: (1 = read beyond the last record; 0 = if not)

FT: is the end of the process: (1 = read beyond the last record; 0 = if not)

And following the statement given to the PAF extractor .

PUFERR: will contain the recording read thanks to the FETCHER statement. Either the following five fields:

PUFMVT: code of the erroneous transaction

PAFTAB: PAF table code of erroneous transaction

PAFEXT: erroneous transaction

PUFGRE: error gravity

PUFLIE: error label

Or ..

PUFENR: will contain the error to write thanks to the INSERT statement, i.e. the following three fields:

PUFMVT: transaction code ('M', 'C', 'A','B' or 'S')

PAFTAB: PAF table code of the transaction

PAFEXT: transaction made of 9 technical characters followed by the specific part (image of the PAF Table description with all its valued fields).

In all cases,

NUMERR: is the order number of the 'error' record

NUMENR: is the order number of the record to write; this field is to be entered before INSERT statement and is supported by the user.

FILLER: (11 character-long) - unused.

Chapter 5. PAF Implementation for Various Environments

Introduction

The use of the PAF function involves the transformation of SQL requests for access to the VisualAge Pacbase database written in user programs, through the generation of data and VA Pac access subprogram calls in the COBOL source generated from these programs.

The PAF Preprocessor (BVPAFP10) processes the generated programs in order to perform this transformation.

Several methods are available to process the generated programs, using PAF:

- use the GPRP procedure. The VA Pac Generation-Printing procedure is followed by the execution of PAF Preprocessor which processes the whole generated flow.
- use the PPAF procedure. The user may:
  - request this procedure in the Optional Control Cards in front of/in back of program, which are combined with the link-edit compilation JCL;
  - call this procedure after the execution of the standard GPRT procedure, from which the generated flow will be retrieved;
  - use any other methods best suited with the characteristics of the site.

  (See the PPAF section in chapter 'Standard Procedures, subchapter 'GPRT: the Generation and Printing Procedure' in 'the Developer's Procedures' manual).

PAF subprograms for the batch and TP mode are provided at installation.

The PAF DICTIONARY

The PAF relational model is a tabular structure of data organized in columns. These columns are defined by VA Pac Data Elements and the tables are associated with Segments or extension Meta Entities.

A PAF dictionary which contains a representation of tables in the form of VA Pac entities - i.e. Data Elements, Segments and Data Structures, a Segment representing a table, a Data Element representing a column) - is available to you.

You can download this dictionary, in French or English from our Support internet site at the following address:

www-306.ibm.com/software/awdtools/vapacbase/support.html

In the 'VisualAge Pacbase downloads' category, click on 'VisualAge Pacbase downloads in English' or 'Downloads VisualAge Pacbase en français'.

In the 'Download package' table, click on the 'Downlaoad options' column of the 'PAF Dictionary' line.

The PAF dictionary is provided in the form of batch transactions. The introduction of this dictionary in the VisualAge Pacbase database is made via the UPDT update procedure.

To avoid problems of compatibility between the site dictionary and the entities of the PAF dictionary, it is strongly recommended to create an independent network of libraries.

NOTE: The PAF dictionary is not required for an extraction using the PAF module.

It only enables, with its descriptions in the form of Segments, to work on the extraction results in an other program.

Example : you want to extract all the programs to modify the code of the control cards in front of programs.

PGM1 : PAF extractor Select ... PGMDEF, writing of the result in a file.

PGM2 : reading of the result file, modification of the 'Cards in front of program' field. and formatting of the UPDP input.

In this case, the structure of the temporary file will be PG01.

OS/390-CICS Version

Specifics

For online processing, it is recommended to use the EIBTRMID field to identify the users in CONNECT statements (IDENT parameter).

Furthermore, the user can access different databases from the same PAF program, if more than one VA Pac database coexist in the same CICS. The database calling code (4 characters), which is assigned to the BASE parameter in the CONNECT statement, makes it possible to select the database that the user wishes to query.

A typical CONNECT statement under CICS looks like this:

```
EXEC PAF CONNECT C001 TO
     USER = ZC00-CUSR
     PASS = ZC00-CPSW
     LIB = ZC00-CLIB
     SESSION = ZC00-CSES
     NET = ZC00-TVIS
     SIZE = ZC00-LSCR
     IDENT = EIBTRMID
     BASE = ZC00-CBAS
```

The workfile required for PAF to operate online has a DDNAME which is imposed under CICS in the BVPPA form. It must be unique for the whole of programs which access the databases installed on the site.

Example of a PAF Extraction User Program

```
000100 //*************************************************************
000120 //* VISUALAGE PACBASE
000130 //*
001100 //*             - JCL EXAMPLE  -
001200 //*    EXECUTION OF A USER P.A.F. BATCH PROGRAM
001300 //*************************************************************
001400 //PAFBATCH PROC BASE=$BASE,  CODE OF VAPAC DATABASE
001600 //      INDSV='$INDSV',      INDEX OF SYSTEM VSAM FILES
001700 //      INDSN='$INDSN',      INDEX OF SYSTEM NON VSAM FILES
001800 //      INDUV='$INDUV',      INDEX OF USER VSAM FILES
001900 //*:    VSAMCAT='$VCAT',     USER VSAM CATALOG
002000 //*:    SYSTCAT='$SCAT',     SYSTEM VSAM CATALOG
002100 //      STEPLIB=,            USER LIBRARY OF LOAD-MODULES
002200 //      OUT=$OUT             OUTPUT CLASS
003000 //*************************************************************
003100 //MAXKEY  EXEC PGM=IDCAMS
003200 //************************
003300 //*:STEPCAT DD DSN=&VSAMCAT,DISP=SHR
003400 //SYSPRINT DD SYSOUT=&OUT
003500 //SYSPAF   DD DSN=&&SYSPAF,DISP=(NEW,KEEP),
003600 //         SPACE=(CYL,(3,3)),
003700 //         LRECL=1031,RECORG=KS,KEYOFF=0,KEYLEN=12
003800 //MAXKEY   DD DSN=&INDSN..BVPSY(MAXKEY),DISP=SHR
003900 //SYSIN    DD DSN=&INDSN..BVPSY(REPRO999),DISP=SHR
005000 //WITHPAF  EXEC PGM=------
005100 //************************
005200 //STEPLIB  DD DSN=&STEPLIB,DISP=SHR
005202 //         DD DSN=$BCOB,DISP=SHR
005300 //*:STEPCAT  DD DSN=&VSAMCAT,DISP=SHR
005400 //PAC7AN   DD DSN=&INDUV..&BASE.AN,DISP=SHR
005500 //PAC7AR   DD DSN=&INDUV..&BASE.AR,DISP=SHR
005600 //PAC7AE   DD DSN=&INDSV..BVPAE,DISP=SHR
005610 //PACGGN   DD DSN=&INDSV..BVPGN,DISP=SHR
005620 //PACGGR   DD DSN=&INDSV..BVPGR,DISP=SHR
005630 //PACGGU   DD DSN=&INDSV..BVPGU,DISP=SHR
005800 //SYSPAF   DD DSN=&&SYSPAF,DISP=(OLD,KEEP)
005900 //------   DD DSN=---
006000 //------   DD DSN=---
006100 //------   DD DSN=---
```

```
006200 //SYSOUT   DD SYSOUT=&OUT
006300 //SYSUDUMP DD SYSOUT=&OUT
007000 //         PEND
008000 //PAFBATCH EXEC PAFBATCH
```

IMS Version

Specifics

All IMS PAF programs, whether batch or online, must always call the online extractor. The IMS version does not include a batch extractor. It is the online extractor that is used in both batch and online processing.

- 'BVPTPST' --> extractor for all entities except keywords
- 'BVPTPWS' --> extractor for keyword entities only

You must always code the 'SET' statement in the WORKING-STORAGE SECTION of batch programs (and online programs if these are developed with the Batch Function). This statement should be coded as follows:

```
EXEC PAF SET TYPE = 01
END-EXEC
```

In addition, you must define, for your application program, a PSB that includes the PCBs of the following databases:

- it may include the PCBs of the user databases,
- it must include the PCBs of the VA Pac system databases:
  - . AN AR AE LB AY AJ PA and DC
  - . GU GN GR GY GJ
  - . TR WS SV

These PCBs must be defined in the PSB in the following way:

```
                              .
                              .
                              .
PCB     TYPE=DB,DBDNAME=BDAN$BASE,PROCOPT=GOT,KEYLEN=49
SENSEG  NAME=PAC7AN
PCB     TYPE=DB,DBDNAME=BDAR$BASE,PROCOPT=GOT,KEYLEN=08
SENSEG  NAME=PAC7AR
PCB     TYPE=DB,DBDNAME=BVPDAE,PROCOPT=GOT,KEYLEN=12
SENSEG  NAME=PAC7AE
PCB     TYPE=DB,DBDNAME=BVPDLB,PROCOPT=GOT,KEYLEN=23
SENSEG  NAME=PAC7LB
PCB     TYPE=DB,DBDNAME=BDAY$BASE,PROCOPT=GOT,KEYLEN=08
SENSEG  NAME=PAC7AY
PCB     TYPE=DB,DBDNAME=BDAJ$BASE,PROCOPT=GOT,KEYLEN=08
SENSEG  NAME=PAC7AJ
PCB     TYPE=DB,DBDNAME=BDPA$BASE,PROCOPT=A,KEYLEN=37
SENSEG  NAME=PAC7PA
PCB     TYPE=DB,DBDNAME=BDDC$BASE,PROCOPT=GOT,KEYLEN=31
SENSEG  NAME=PAC7DC
```

```
PCB     TYPE=DB,DBDNAME=BVPDGU,PROCOPT=GOT,KEYLEN=8
SENSEG  NAME=PACGGU
PCB     TYPE=DB,DBDNAME=BVPDGN,PROCOPT=GOT,KEYLEN=49
SENSEG  NAME=PACGGN
PCB     TYPE=DB,DBDNAME=BVPDGR,PROCOPT=GOT,KEYLEN=08
SENSEG  NAME=PACGGR
PCB     TYPE=DB,DBDNAME=BVPDGY,PROCOPT=GOT,KEYLEN=8
SENSEG NAME=PACGGY
PCB    TYPE=DB,DBDNAME=BVPDGJ,PROCOPT=GOT,KEYLEN=08
SENSEG NAME=PACGGJ
PCB    TYPE=DB,DBDNAME=BVPDTR,PROCOPT=GOT,KEYLEN=08
SENSEG NAME=PAC7TR
PCB    TYPE=DB,DBDNAME=BVPDWS,PROCOPT=GOT,KEYLEN=37
SENSEG NAME=PAC7WS
PCB    TYPE=DB,DBDNAME=BVPDSV,PROCOPT=GOT,KEYLEN=15
SENSEG NAME=PAC7SV
PSBGEN  PSBNAME=psbname,LANG=COBOL
END
                              .
                              .
```

where $BASE = Database code chosen when VisualAge Pacbase was installed.

Important note: Extractor call statements, which are generated by the preprocessor, have the following format:

CALL 'extractor' USING S-PCB-AN S-PCB-AR S-PCB-AE S-PCB-LB S-PCB-AY S-PCB-AJ S-PCB-PA

S-PCB-DC S-PCB-GU S-PCB-GN S-PCB-GR S-PCB-GY S-PCB-GJ

S-PCB-TR S-PCB-WS S-PCB-SV

cursor-name.

Therefore, you must give the same names to PCBs in the LINKAGE SECTION and the PROCEDURE DIVISION USING in batch Programs (S-PCB-xx). For OLSD Screens, the definition screens of the PCBs called in the dialog PSB must be named PCB-xx. The OLSD generator will add the prefix 'S-' in the source generated on the LINKAGE SECTION and PROCEDURE DIVISION USING level.

EXTRACTION UNDER THE CONTROL OF A SECURITY SYSTEM

Entities can be extracted under the control of a security system (e.g., RACF). In this case, the extractor must be able to tell whether the program doing the extraction is batch or online. The method for controlling the user code is actually different for a batch or an online program.

In batch processing, the user code, given in the CONNECT statement, is directly controlled, in relation to the security system, by means of an assembler program, PACSECB, which is transparent to the user.

In online processing, the user code is controlled, in relation to the one listed in the IO-PCB, by the security system.

In order to achieve this control, the extractor must know the type of the program calling it (batch or online). To do this, the MODE parameter must be coded in the SET order:

```
- MODE = 'TP   ' <- the user program is an online
                    program
- MODE = 'BATCH' <- the user program is a batch program
```

Important note: For an online program, extractor call statements, which are generated by the preprocessor, will appear as follows:

CALL 'extractor' USING S-PCB-AN S-PCB-AR S-PCB-AE S-PCB-LB S-PCB-AY S-PCB-AJ S-PCB-PA

S-PCB-DC S-PCB-GU S-PCB-GN S-PCB-GR S-PCB-GY S-PCB-GJ

S-PCB-TR S-PCB-WS S-PCB-SV

cursor-name S-IPCB.

For batch programs, call orders are generated the same way, with or without the control of a security system (without the S-IPCB parameter in the CALL).

Example of a PAF Extraction User Program

```
000100 //*************************************************************
000120 //* VISUALAGE PACBASE
000130 //*
001100 //* INSTALLATION :    - JCL EXAMPLE  -
001200 //*    EXECUTION OF A USER P.A.F. BATCH PROGRAM
001300 //*************************************************************
001400 //PAFBATCH PROC BASE=$BASE,  CODE OF VAPAC DATABASE
001600 //         INDSV='$INDSV',      INDEX OF SYSTEM VSAM FILES
001700 //         INDSN='$INDSN',      INDEX OF SYSTEM NON VSAM FILES
001800 //         INDUV='$INDUV',      INDEX OF USER VSAM FILES
001900 //*:       VSAMCAT='$VCAT',     USER VSAM CATALOG
002000 //*:       SYSTCAT='$SCAT',     SYSTEM VSAM CATALOG
002100 //         STEPLIB=,            USER LIBRARY OF LOAD-MODULES
002200 //         OUT=$OUT,             OUTPUT CLASS
002300 //         PSBLIB='$PSBLIB',    IMS PSBLIB
002400 //         DBDLIB='$DBDLIB',    IMS DBDLIB
002500 //         RESLIB='$RESLIB',    IMS RESLIB
002600 //         PROCLIB='$PRCLIB',   IMS WORKLIB
002700 //        BUF=40,SPIE=0,TEST=0,EXCPVR=0,RST=0,PRLD=,SRCH=0,
```

```
002800 //        CKPTID=,MON=N,LOGA=0,FMTO=T,DBRC=$DBRC,IRLM=$IRLM
003000 //*************************************************************
003100 //MAXKEY  EXEC PGM=IDCAMS
003200 //***********************
003300 //*:STEPCAT DD DSN=&VSAMCAT,DISP=SHR
003400 //SYSPRINT DD SYSOUT=&OUT
003500 //SYSPAF   DD DSN=&&SYSPAF,DISP=(NEW,KEEP),
003600 //         SPACE=(CYL,(3,3)),
003700 //         LRECL=468,RECORG=KS,KEYOFF=0,KEYLEN=12
003800 //MAXKEY   DD DSN=&INDSN..BVPSY(MAXKEY),DISP=SHR
003900 //SYSIN    DD DSN=&INDSN..BVPSY(REPRO999),DISP=SHR
005000 //WITHPAF  EXEC PGM=DFSRRC00,REGION=$REGSIZ,
005050 //         EXEC PARM=(BMP,------, ... ETC ...
005060 //.....
005100 //.....
005200 //STEPLIB  DD DSN=&RESLIB,DISP=SHR
005220 //         DD DSN=&STEPLIB,DISP=SHR
005250 //         DD DSN=$BCOB,DISP=SHR
005300 //...
005400 //B7AN$BASE   DD DSN=&INDUV..&BASE.AN,DISP=SHR
005500 //B7AR$BASE   DD DSN=&INDUV..&BASE.AR,DISP=SHR
005600 //BVP7AE   DD DSN=&INDSV..BVPAE,DISP=SHR
005610 //BVP7GN   DD DSN=&INDSV..BVPGN,DISP=SHR
005620 //BVP7GR   DD DSN=&INDSV..BVPGR,DISP=SHR
005630 //BVP7GU   DD DSN=&INDSV..BVPGU,DISP=SHR
005800 //B7PA$BASE   DD DSN=&INDUV..&BASE.PA,DISP=SHR
005900 //B7P1$BASE   DD DSN=&INDUV..&BASE.P1,DISP=SHR
006000 //------   DD DSN=---
006100 //------   DD DSN=---
006200 //*
006300 //
```

Windows/NT Version

Specifics

The pre-processor is made up of a program installed in the directory of the VA Pac server programs, '\SYS\PGM'.

THE EXTRACTION SUBPROGRAMS

The PAF user programs (batch or online), generated with a '3' variant (to comply with Micro Focus COBOL), use the same extraction subprograms.

There are three extraction subprograms:

- BVPBBTST (for standard extractions) and BVPBBTWS (for keyword extractions) are called dynamically by PAF user programs.
- BVPBFILE is dynamically called by the extractors (BVPBBTST or BVPBBTWS) to access the VA Pac Database and the PAF workfile.

The three subprograms, which are called dynamically, are supplied, compiled and linked (.DLL files) as COBOL source (.CBL files). The .DLL files are installed in the batch programs' directory, '\SYS\PGM', VA Pac servers.

Example of compilation: CBLLINK -D [program_name].CBL

The COBOL source files of the extractors are supplied in the PAFCBL directory.

The extractors which are dynamically called must be compiled and linked at the site when the version of the site's Micro Focus compiler is not compatible with that used for the VA Pac release.

The following list gives the compilation instructions that must be used to compile the extraction subprograms (Micro Focus Net Express V3) :

```
*          - VERBOSE
*          - NOANIM
*          - ASSIGN"EXTERNAL"
*          - NOBOUND
*          - IBMCOMPDIR
*          - NOLIST
*          - NATIVE"EBCDIC"
*          - VSC2E
*          - PERFORM-TYPE"OSVS"
*          - SEQUENTIAL"LINE"
*          - LINKCOUNT"96"
*          - NOTRACE
*          - NOWARNING
*          - CALLFH"FHREDIR"
*          - NOTRUNC
*          - NOCOBOLDIR
*          - NOBELL
*          - INITCALL"UTILS"
```

EXAMPLE OF COMPILATION AND LINK OF A PAF PROGRAM

The purpose is to compile a batch program.

The compiler used for VA Pac programs is Micro Focus Net Express V3. It is advised to use the same compiler version for the PAF programs to avoid conflicts between Micro Focus libraries.

Compilation instructions:

```
ASSIGN "EXTERNAL"
SEQUENTIAL "LINE"
```

Compilation and link command:

```
CBLLINK -E [program_name].CBL
```

EXAMPLE OF AN EXECUTION PROCEDURE USING PAF

You want to execute a batch program.

```
*** Delete previous PAF files ******
*** Assignment of the database files and PAF ***
*** Assignment of user files ***
 (Add user program's specific files)
*** Execution of the program
*** End
*** Management of execution error, BVPAFP10.EXE.
```

Example of a PAF Extraction User Program

```
000100 REM * -------------------------------------------------------
000200 REM *      VISUALAGE PACBASE
000300 REM *
000400 REM * -------------------------------------------------------
000500 REM *         - EXAMPLE OF PAF PROCEDURE EXECUTION -
000600 REM *
000700 REM * -------------------------------------------------------
000800
000900 Set WshShell = Wscript.CreateObject("Wscript.Shell")
001000 Set Args = Wscript.Arguments
001100 Set WshEnv = WshShell.Environment("PROCESS")
001200 Set WshVolEnv = WshShell.Environment("VOLATILE")
001300
001400 Dim FSO
001500 Set  FSO = CreateObject("Scripting.FileSystemObject")
001600 Dim ObjProcess
001700 Set  ObjProcess = CreateObject("BvpWsh.Process")
001800
001900 Rep_BVP="HKLM\SOFTWARE\IBM\BVP VisualAge Pacbase\Server\BVP"
002000
002100 Rep_Proc = WshShell.RegRead(Rep_BVP & "_SYS\PROC\")
002200 Rep_NLS = WshShell.RegRead (Rep_BVP & "_SYS\NLS\")
002300 Rep_SKEL = WshShell.RegRead (Rep_BVP & "_SYS\SKEL\")
002400 Rep_PGM = WshShell.RegRead (Rep_BVP & "_SYS\PGM\")
002500
002600   WshEnv("PATH") = Rep_PGM & ";" & WshEnv("PATH")
002700
002800 ' Set COBOL environments :
002900 ' ------------------------
003000 Var = "HKLM\SOFTWARE\Micro Focus\NetExpress\"
003100
003200 On Error Resume Next
```

```
003300 NetExpress_Version = WshShell.RegRead (Var & _
003400                        "DefaultVersion")
003500 If Err.Number = 0 Then
003600    NetExpress = 1
003700 Else
003800    NetExpress = 0
003900 End if
004000
004100 On Error Resume Next
004200 NetExpress_Version = WshShell.RegRead (Var & "Version")
004300 If Err.Number = 0 Then
004400    ApplicationServer = 1
004500 Else
004600    ApplicationServer = 0
004700 End if
004800
004900
005000 If NetExpress = 1 or ApplicationServer = 1 Then
005100    Var = Var & NetExpress_Version & "\COBOL\"
005200    COBOL_Version = WshShell.RegRead (Var & "Version")
005300    Var = Var & COBOL_Version & "\Environment\"
005400    COBPATH =  WshShell.RegRead (Var & "PATH")
005500
005600    WshEnv("COBDIR") = WshShell.RegRead (Var & "COBDIR")
005700    WshEnv("COBHNF") = WshShell.RegRead (Var & "COBHNF")
005800    WshEnv("PATH") = COBPATH & ";" & WshEnv("PATH")
005900 Else
006000  wscript.echo "NetExpress or Application Server
006100                              not implemented. "
006200 End if
006300 '============================================================
006400
006500 ' EXECUTION OF PAFPGM
006600 '----------------------------------
006700
006800 ' 1-Delete PAF File
006900 If  FSO.FileExists(Rep_TMP & "\WPAF.tmp") Then
007000 Set FileD = FSO.GetFile(Rep_TMP & "\WPAF.tmp")
007100 FileD.Delete
007200 End If
007300
007400 ' 2-Assign database files ...
007500 WshEnv("PAC7AE") = Rep_SKEL & "\AE"
007600 WshEnv("PAC7AR") = Rep_BASE & "\AR"
007700 WshEnv("PAC7AN") = Rep_BASE & "\AN"
007800 WshEnv("PACGGN") = Rep_ABASE & "\AN"
007810 WshEnv("PACGGR") = Rep_ABASE & "\AR"
007820 WshEnv("PACGGU") = Rep_ABASE & "\GU"
007840
007900 ' 2bis- ... and  PAF File
008000 WshEnv("SYSPAF") = Rep_TMP & "\WPAF.tmp"
008100
008200 ' 3-Assign of program  files
008300 ' ......................
008400
```

```
008500 ' 4- Execute ...
008600 Return = WshShell.Run("BVPAFPGM.exe ", 1, TRUE)
008700
008800 ' 5- If Error
008900 if err.number <> 0 and Return = 0 then
009000  wscript.echo "Syntax or RunTime Error"
009100 End if
009200
009300 if  Return <> 0 then
009400  wscript.echo "PAFPGM execution error"
009500 End if
009600
009700
009800 Wscript.Quit (Return)
009900
```

UNIX Version

Specifics

EXTRACTION SUBPROGRAMS

For user programs generated with variant '3' of TYPE OF COBOL TO GENERATE (adaptation to Micro Focus COBOL), the same extraction subprograms are used for both batch and online processing. These extraction subprograms are supplied, compiled and linked (.int and .gnt files). .int ou .gnt).

There are three extraction subprograms:

- BVPBBTST (for standard extractions) BVPBBTWS (for keyword extractions) are dynamically called by the PAF user programs.
- BVPBFILE is dynamically called by the extractors (BVPBBTST or BVPBBTWS) for access to the VisualAge Pacbase Database and to the PAF workfile.

  These subprograms must be compiled on the site when the release of the site Micro Focus is different from the one used for VisualAge for UNIX.

  The release of the Micro Focus compiler used for VisualAge Pacbase for UNIX is:

- MicroFocus 4.1.10 for the TRUE64 platform,
- MicroFocus HP 11.30 for the HP9000 platform,
- MicroFocus Objet Cobol 4.1.30 for the LINUX platform,
- MicroFocus Objet Cobol 4.1.30 for the SUN platform,
- MicroFocus Server Express 2.0.10 for the AIX platform.

COMPILATION OF PAF PROGRAMS

The compilation is executed from the .int files supplied in the $PACDIR/system/int directory.

Example of a compilation script for PAF programs:

```
.!/bin/sh

. Assignment of the COBOL compiler:

COBDIR=/usr/lib/cobol

export COBDIR

. Assignment of LIBPATH or LD_LIBRARY_PATH

LIBPATH=$COBDIR/lib:$LIBPATH

export LIBPATH

# Assignment of the PATH:

PATH=$COBDIR/bin:$PATH

export PATH

. List of programs to be compiled:

PGM="BVPBBTST.int BVPBBTWS.int BVPBFILE.int"

. Compilation start-up:

cob -uv $PGM $COBOPT

. Output files: BVPBBTST.gnt BVPBBTWS.gnt BVPBBT98.gn .gnt
```

Example of a compilation script for user PAF programs:

```
.!/bin/sh

. Assignment of the cobol compiler:

COBDIR=/usr/lib/cobol

export COBDIR

. Assignment of LIBPATH or LD_LIBRARY_PATH
```

LIBPATH=$COBDIR/lib:$LIBPATH

export LIBPATH

Assignment of PATH:

PATH=$COBDIR/bin:$PATH

export PATH

Compilation instructions:

COBOPT="-C ASSIGN=EXTERNAL -C NATIVE=EBCDIC -C
SEQUENTIAL=LI

COBOPT="$COBOPT -C PERFORM-TYPE=OSVS -C OSVS -C NOBOUND -C
I

COBOPT="$COBOPT -C NESTCALL -C DEFAULTBYTE=32"

List of programs to be compiled:

PGM="PGPAF.cbl PGPAFP.cbl"

Compilation start-up:

cob -uv $PGM $COBOPT

Output files: PGPAF.gnt and PGPAFP.gnt

EXECUTION OF A PAF EXTRACTOR

Before executing the PAF extractor, you should perform the following file assignments:

```
. Permanent input files:
  - VisualAge Pacbase data file for UNIX  : PAC7AR
  - VisualAge Pacbase index file for UNIX : PAC7AN
  - Error message file                    : PAC7AE
. PAF workfile                            : SYSPAF
.  User files (when necessary).
```

Example of a PAF Extraction User Program

```
000100 #!/bin/sh
000200 #
000300 # EXEMPLE OF AN EXECUTION PROCEDURE USING A PAF PROGRAM
000400 # ------------------------------------------------------
000500 # (execution of PGPAF program)
```

```
000600
000700 # Cobol compiler environment variable :
000800 COBDIR=/usr/local/cobol
000900 export COBDIR
001000
001100 # PATH environment variable :
001200 PATH=.:$COBDIR/bin:$PATH
001300 export PATH
001400
001500 # LIBPATH environment variable :
001600 # (for AIX version)
001700 LIBPATH=/usr/lib:$COBDIR/lib
001800 export LIBPATH
001900
002000 # LD_LIBRARY_PATH environment variable :
002100 # (for HP-UX, OSF1, SunOS or Linux version)
002200 LD_LIBRARY_PATH=/usr/lib:$COBDIR/lib
002300 export LD_LIBRARY_PATH
002400
002500 # VisualAge Pacbase directory environment variable :
002600 PACDIR="/lv00/ll350/paclanx"
002700 export PACDIR
002800
002900 # COBPATH environment variable :
003000 # (current directory + VisualAge Pacbase programs directory)
003100 COBPATH=.:$PACDIR/system/gnt
003200 export COBPATH
003300
003400 # VisualAge Pacbase database name (mandatory) :
003500 BASE=BVAP
003600
003700 # VisualAge Pacbase environment variable (mandatory) :
003800 . $PACDIR/config/$BASE/PAC7AE.ini
003900 . $PACDIR/config/$BASE/PAC7AN.ini
004000 . $PACDIR/config/$BASE/PAC7AR.ini
004100 . $PACDIR/config/$BASE/PACGGN.ini
004200 . $PACDIR/config/$BASE/PACGGR.ini
004300 . $PACDIR/config/$BASE/PACGGU.ini
004400 SYSPAF=./wpaf
004500 export SYSPAF
004600
004700 # User application environment variable :
004800 FILE1=./file1
004900 export FILE1
005000
005100 # Start execution :
005200 cobrun PGPAF
005300
005400 # Deletion of the temporary files :
005500 if [ -r "$SYSPAF" ]
005600 then
005700    rm $SYSPAF*
005800 fi
```

Chapter 6. Error Messages

The PAF Translator

The PAF Translator can detect a number of syntax errors in SQL-PAF statements. Each error, including the corresponding error message and the line number which identifies the beginning of the PAF sequence in the translated program, is printed in an output report.

The possible error messages are listed below, including explanatory comments, in some cases.

UNKNOWN COLUMN CODE : <column-code>

The <column-code> does not identify a table column specified in the FROM clause (in the selected language).

TOO MANY ELEMENTARY CONDITIONS IN SELECT CLAUSE

There are more than 50 elementary conditions in this SQL-PAF query.

CURSOR CODE IS TOO LONG : <cursor-code>

The cursor code must contain four characters.

CURSOR CODE ALREADY DECLARED : <cursor-code>

TOO MANY CURSORS DECLARED

There are more than 100 cursors declared in this SQL-PAF query.

UNKNOWN CURSOR CODE : <cursor-code>

There is a cursor management statement for a cursor which has not been declared in the PAF user program.

NO CONNECT STATEMENT FOR CURSOR : <cursor-code>

NO OPEN STATEMENT FOR CURSOR : <cursor-code>

NO FETCH STATEMENT FOR CURSOR : <cursor-code>

NO CLOSE STATEMENT FOR CURSOR : <cursor-code>

NO INIT STATEMENT FOR CURSOR : <cursor-code>

THE PAF SEQUENCE IS TOO LONG

A PAF sequence is a series of lines grouped between EXEC PAF and END-EXEC. The maximum number of these lines is 50.

END OF PROGRAM DURING A PAF SEQUENCE

OPERAND CANNOT BE NUMERIC : <operand>

OPERAND CANNOT BE ALPHANUMERIC : <operand>

INVALID OPERAND LENGTH : <operand>

Alphanumeric constant operands have a maximum length of 120 characters.

INVALID COBOL OPERAND : <operand>

COLUMN TYPES ARE DIFFERENT : <col1-code> <col2-code>

An elementary condition applies in the comparison of a numeric column with an alphanumeric column.

LEFT PARENTHESIS MISSING

RIGHT PARENTHESIS MISSING

Elementary conditions which follow the keyword WHERE must be enclosed between balanced parentheses, i.e., the number of LEFT parentheses must equal the number of RIGHT parentheses.

NO QUIT STATEMENT IN PAF-USER PROGRAM

SYNTAX ERROR : <erroneous-syntax>

Incorrect syntax for the SQL-PAF language.

INVALID LITERAL LENGTH : <literal>

The maximum length of a literal is 120 characters.

TOO MANY LITERALS ON A SINGLE LINE

The number of literals on a PAF sequence line must not exceed 40.

INCORRECT ENDING OF LITERAL : <literal>

UNKNOWN TABLE CODE : <table-code>

The <table-code> does not identify a PAF table (in the selected language).

UNKNOWN UE TYPE CODE (WRONG TABLE CODE): <UE-type-code>

The User Entity Table code is incorrect because the UE Type code, used to build the generic Table code, does not exist (in the selected sub-network).

For more details, refer to the description of the User Entity Tables.

UNKNOWN UE DESCRIPTION (WRONG TABLE CODE) : <Dxx>

The code of the User Entity Table is incorrect since the specified Meta-Entity Description Number does not exist.

For more details, refer to the description of the User Entity Tables.

INVALID PACBASE CONNECTION PARAMETERS

INVALID STRING DELIMITER : <delimiter>

The delimiter specified in the SET statement must have a value of either SINGLE (single (') quotes) or DOUBLE (double (") quotes), respectively.

INVALID EXECUTION MODE : <execution-mode>

The execution mode specified in the SET statement must have a value of either BATCH or TP.

INVALID GENERATION VARIANT(S) : <generation-variant(s)>

The generation variant(s) specified in the SET statement must be VisualAge Pacbase variant(s).

THERE SHOULD NOT BE A CONDITION ON COL.: <column-number>

There can't be any conditions on columns 05 (VA Pac entity code), 06 (VA Pac entity label), and 07 (explicit entity keywords) of the KEYWORD table.

THERE SHOULD NOT BE SEVERAL CONDITIONS ON COL. 01

There can't be more than one condition on column 01 (entity type) of the KEYWORD table.

THERE SHOULD NOT BE SEVERAL CONDITIONS ON COL. 02

There can't be more than one condition on column 02 (UE type code) of the KEYWORD table.

THERE SHOULD NOT BE SEVERAL CONDITIONS ON COL. 03

There can't be more than one condition on column 03 (keyword type) of the KEYWORD table.

THERE SHOULD NOT BE SEVERAL CONDITIONS ON COL. 04

There must be one and only one condition on column 04 (WS search argument).

THERE SHOULD BE A CONDITION ON COLUMN 01

If there is an elementary condition on column 02 (UE type code), then there must be a condition on column 01 (entity type).

INCORRECT COMPARATOR FOR COLUMN: <column-number>

Only the '=' operator can be used in the elementary conditions of a KEYWORD table query.

INCORRECT OPERAND FOR COLUMN: <column-number>

The operand in the elementary condition of a KEYWORD table query mustn't be another column of the table.

ACCESS TO PAF IS NOT ALLOWED

Check the access key.

The PAF Extractor

ERROR CODES RETURNED BY THE EXTRACTOR SUBPROGRAM

The <cursor-code>-RETCOD field, generated by the PAF Translator, contains the error code returned by the PAF Extractor subprogram.

The '00' value in this field indicates that no error was detected.

There are three types of errors:

- SQL-PAF statement sequence errors,

- File access errors,
- Errors in extracted data.

SQL-PAF STATEMENT SEQUENCE ERRORS

Return Codes : 01 to 10

The following chart summarizes the errors which can occur in the sequence of SQL-PAF statements.

Statements on lines precede statements in columns.

"NULL" means that no prior statement has been issued.

When there is a sequence error, the corresponding box contains the return code value.

EXAMPLE: The first line indicates that no statement can be issued before the INIT statement.

	INIT	**CONNECT**	**OPEN**	**FETCH**	**CLOSE**	**QUIT**
NULL		01	01	01	01	01
INIT	02		03	04	05	
CONNECT	02			06	07	
OPEN	02		08			
FETCH	02		08			
CLOSE	02			09	10	
QUIT		01	01	01	01	01

RETURN CODE VALUES AND MEANING

```
01   Initializations not performed.
02   Initializations already performed.
03   OPEN    of an unconnected cursor.
04   FETCH   of an unconnected cursor.
05   CLOSE   of an unconnected cursor.
06   FETCH   of an unopened cursor.
07   CLOSE   of an unopened cursor.
08   OPEN    of an unclosed cursor.
09   FETCH   of a closed cursor.
10   CLOSE   of a closed cursor.
```

FILE ACCESS ERRORS

File access errors occur in relation to the VisualAge Pacbase Database files (Index, Data, and Error Messages) and the Temporary Workfile.

RETURN CODE VALUES AND MEANING

21: Open error on Index File,

22: Open error on Data File,

23: Open error on Error Message File,

24: Open error on Temporary Workfile,

31: Read/Write error on Temporary Workfile,

32: Read error on VisualAge Pacbase File,

40: VisualAge Pacbase Database Connection error.

41: Unauthorized use of PAF (access key).

45: Unknowm user code

46: Unavailable password

47: Erroneous application code

48: Invalid session

49: Invalid absence of user code

ERRORS IN EXTRACTED DATA

Return Code: 50

Errors in extracted data occur with numeric columns in User-Entity Tables. This happens when the internal format of a Data Element which describes a Meta-Entity can be modified even though User Entity of that Meta-Entity have already been created. Thus, the content of the column associated with this Data Element can be alphanumeric instead of numeric.

In such cases, the query is still valid. It is when the FETCH statement is issued that the Extractor returns error code '50'.

Chapter 7. Presentation of the PAF-PDM Functions

Foreword

The PAF Function and the PDM Extension support functions which may be used jointly.

They do not replace the initial PAF and PDM functions but enhance them as they co-operate.

Hereafter, they will be referred to as PAF-PDM functions.

NOTE: The PAF-PDM functions may also be used independently of each other.

These functions are therefore sub-divided into PAF+ and PDM+.

The following page lists all the manuals and documents which may be necessary when using the PAF-PDM functions.

Using PAF-PDM requires an in-depth knowledge of the VisualAge Pacbase metamodel and (if installed) the metamodels of the WorkStation's Pacdesign or Pacbench modules (specific to the methodology in use).

DOCUMENTATION

Listed below is the exhaustive list of manuals and documents which may be necessary when using PAF-PDM:

1. PAF Reference Manual, with an appendix containing two examples of Extraction Master Paths including the execution reports printed by the XPAF Validation procedure.
2. The description of PAF tables.

   The Documentation of the PAF Tables is available at the following address:

   www-1.ibm.com/support/docview.wss?rs=37&uid=swg27005477.

   Click on the 'PAF module' choice. The links you find there lead you to the lists of the PAF tables of all the VA Pac entities, sorted by functional unit:

   - the entities of the standard VA Pac metamodel,
   - the entities of the eBusiness metamodel,
   - the entities managed in the Administration database and the Administration management entities linked to a Development database,
   - the Methodology entities.

A cumulative index of all the tables for all the entities is also available at this address. This index enables you to branch directly to the description of a selected table.

3. VisualAge Pacbase Procedures Manuals.

Objectives of PAF-PDM Functions

AUTOMATIC STRUCTURING AND MAINTENANCE OF VOLUMES

The initial purpose of PAF-PDM is to add new functionalities to the PDM extension.

The underlying principle of these functionalities is to make the most of the cross-references between entities in the metamodel.

EXAMPLE:

You want to generate the comments on a Screen instance, i.e the comments entered in:

- the Comments (-GC) of this Screen,
- the Comments (-GC) of its Segments,
- the Description (-D) of the Data Elements called by these Segments.

With PDM, you have to write each individual Segment call into the Volume Description (-D).

As a result, when a new Segment is called in the Screen (-CS), it must also be added in the Volume Description.

With PAF-PDM, you specify the information to be printed in the Volume by defining -- only once -- an Extraction Master Path, also called PTEx.

In the example shown above, the extraction path will start with the Screen entity, find the called Segments, and finally work its way down to the Data Elements, its course being guided by the VA Pac Metamodel.

PAF-PDM is therefore a tool not only for the automatic documentation of applications, but also for the automatic maintenance of this documentation. When the documented application is modified, you only have to re-generate the relevant Volumes without having to change their Description.

DOCUMENTATION STANDARDIZATION

PDM+ allows you to write Master Outlines (PTEds), i.e. skeletons which may be used for various purposes:

- With PDM, print options assigned to a Volume apply to this particular Volume only. As a result, options must be specified in each Volume which makes standardization not an easy task.

PDM+ allows you to specify all relevant print options in one Master Outline. This PTEd is then called by as many Volumes as needed.

- Coding standardized calls is another PDM+ facility.

For instance, in a Master Outline, the following call:

```
TGEN___D__
```

will cause all Text instances whose code starts with the letters 'GEN' to be printed in ALL Volumes where this Master Outline is invoked.

- Furthermore and most important, standardization in documentation structuring is achieved with PDM+, in co-operation with PAF+.

It is in this framework that the expression Master Outline takes on its full meaning as the PTEd becomes a structural skeleton. The generation of different Volumes in relation to several applications may be based on only one Master Outline (PTEd) managing data extracted by one PTEx.

NOTE: A Volume may call several Master Outlines.

CONCLUSION

With PAF-PDM, automatic documentation structuring and standardization are not synonyms of strictness and rigidity since data extraction and printing are completely user-defined.

However, this definition should come from one authoritative entity, failing which standardization may prove a vain word.

Operating Mode of PAF-PDM Functions

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

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

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

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

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

PAF+: THE EXTRACTION MASTER PATH

PAF+ allows you to write an Extraction Master Path (PTEx), i.e. an exploration course throughout the Specifications Database, from which a data extraction program is automatically generated.

The writing of a PTEx means defining and describing a User Entity of a dedicated Meta-Entity coded '.PPTEX' and whose type code is 7E (CH: Y7E.....).

There are two types of PTEx, therefore the User Entities of the '.PPTEX' Meta-Entity may be of either one of the following types:

- Type 'M' allows you to generate a Macro-Command, i.e. a subprogram which will have to be called in a Master Outline (PTEd).
- Type 'E' allows you to generate a User Extractor program executed independently.

The input of an Extraction Master Path User Entity is documented in the PAF Reference Manual, chapter 'Extraction Master Path: Definition / Description'.

VALIDATION

The Extraction Master Path must then be validated by the XPAF batch procedure which generates the User Extraction Program or the Macro-Command subprogram.

>>>>: The XPAF procedure is documented in the 'Developer's Procedures' Manual, chapter 'Personalized Extraction / Automated Documentation' and in the 'Administrator's Procedures' Manual, chapter 'Manager's Utilities', subchapter 'PACX - Extractions'.

When no error is detected, the validation produces a COBOL source program which must be compiled and linked to be executed.

Execution of a user extractor (E-type PTEx)

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

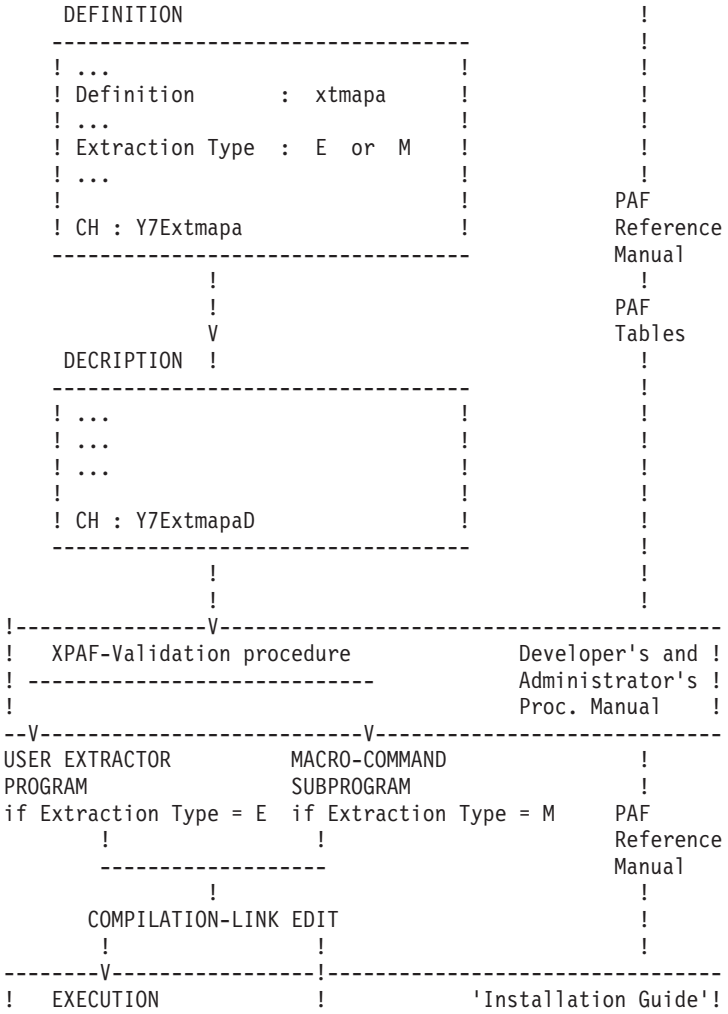
Execution of a macro-command (M-type PTEx):

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

Note

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

PAF+: EXTRACTION MASTER PATH - DESCRIPTION OF STEPS



```
--------V-----------------V---------------------------------
 DATA EXTRACTED          A Macro-Command must be called
                         by a PTEd. See PDM+ Functionality
```

PDM+: THE MASTER OUTLINE

PDM+ allows to write Master Outlines supported by instances of the P-type Volume entity.

A Master Outline organizes the printing of data extracted by PAF+ when its Description calls an Extraction Master Path (on an M-type line).

Master Outlines are reusable as they may be called in different Volumes where they can be customized.

****

Once defined and described a Master Outline must be validated by the XPDM procedure.

>>>>: The XPDM procedure is documented in the 'Developer's Procedures' Manual, chapter 'Personalized Extraction / Automated Documentation'.

****

NOTE: A Master Outline cannot be printed per se. It needs be called in another Volume which when printed will include the Master Outline contents.

However, it is possible to print the Description of a Master Outline (GPRT procedure, DCV command).

>>>: You may also use the PRGS procedure which prints the file used to store Extraction Master Paths and their calling Master Outlines.

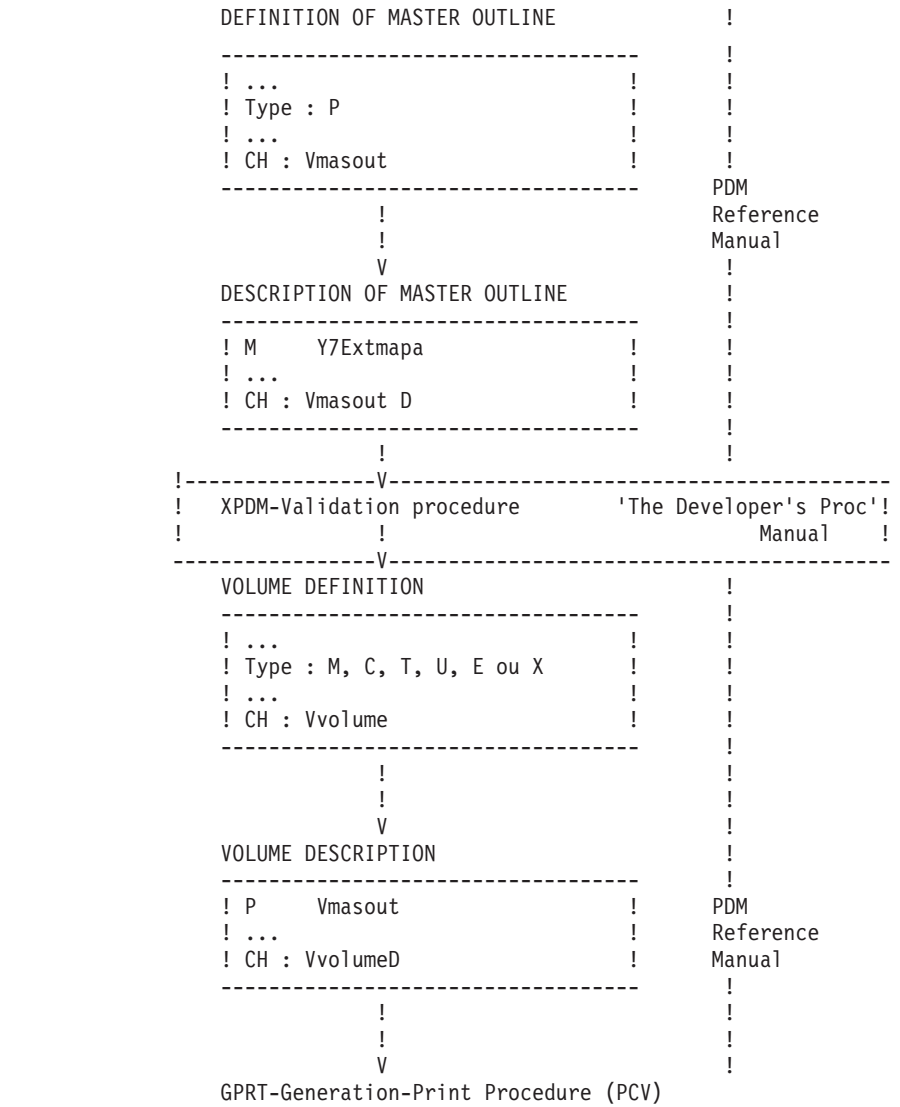
>>>: The PRGS procedure is documented in the 'Developer's Procedures' Manual, chapter 'Personalized Extraction / Automated Documentation'.

A Master Outline is independent of the VA Pac Database in which it is defined and described, provided the Databases roots are identical.

If an Extraction Master Path is modified and then re-validated (XPAF), all Master Outlines calling this Extraction Master Path may have to be modified and must always be re-validated (XPDM).

PDM+: THE MASTER OUTLINE - DESCRIPTION OF STEPS

The chart below presents PAF-PDM used jointly: the Macro-Command subprogram is called in a Master Outline.



Chapter 8. Extraction Master Path: Definition / Description

An Extraction Master Path is defined and described on a User Entity which is an instance of a Meta-Entity dedicated to PAF+/Extraction. The code of this Meta-Entity is '.PPTEX' and its call type is '7E'. It is supplied on installation and must not be modified.

The following pages explain the Definition screens (CH: Y7E.....) and Description screens (CH: Y7E.....D) of the User Entities. A brief description of the input fields is also included.

The values given in the definition and the description of the User Entities are not checked upon user input. For example, if a field that is listed as required in the following pages is not filled in, the user will not be given an error message.

It is advised to document the Extraction Master Path by drawing an extraction tree on the Comments screen (-GC) of the User Entity and by writing comments on the Description screen. Comments will automatically be printed when the validation procedure of the Extraction Master is submitted.

```
Definition                    TYPE : 7E     META-ENTITY :.PPTEX
User entity code   : 1_____
User entity name   : 2____________________________________
Program code       : 3_______
Extraction type    : 4
Options            : 5__________________________________________________
Max. record size   : 6__
Sort option        : 7____
```

NU	LE	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	6		User entity code (ALPHABETIC) IDENTIFIER DATA ELEMENT INVALID ABSENCE UPPERCASE
2	36		User entity name (ALPHABETIC) ENTITY LABEL INVALID ABSENCE
3	8		Program code (ALPHABETIC) INVALID ABSENCE UPPERCASE
			The value input in this field indicates where the COBOL source of the Extraction Master Path will be stored before its compilation, and what the PROGRAM-ID clause contains.
			Furthermore if the Extraction Master Path is a user extraction program, this value is the external name of the executed program.
4	1		Extraction type (ALPHABETIC) INVALID ABSENCE UPPERCASE
		M	Macro-command (subprogram)
		E	User extraction program (program)
5	50		Options (ALPHABETIC)
		STATI	Default value: Static CALL of the PAF extractor
		DYNAM	Dynamic CALL of the PAF extractor
6	3		Max. record size (NUMERIC)
			maximum size of the extracted records formatted by presentation lines. This field only applies to a User Extraction ('E' extraction type).
7	5		Sort option (ALPHABETIC) UPPERCASE
			There are two sort options regarding the Extractor's output results:
		CURS	Default value: Primary sort on the extracted occurrence (entity type and occurrence code). Secondary sort on the identification criterion. (extraction pathway which leads to an occurrence)
		IDENT	Primary sort on the identification criterion. Secondary sort on the extracted occurrence (entity type and occurrence code)
			For a macro-command, the CURS option is always implemented.

```
Description                    7E ______                              01
                                ..................................
Lin L T REF. SELECTION-COMMENT                          Virtual  Proc
___ _ _ ____ ____________________________________________ ________ ____
1__ 2 3 4___ 5_____________________________________________ 6_______ 7___
___ _ _ ____ _____________________________________________ ________ ____
___ _ _ ____ _____________________________________________ ________ ____
___ _ _ ____ _____________________________________________ ________ ____
___ _ _ ____ _____________________________________________ ________ ____
___ _ _ ____ _____________________________________________ ________ ____
___ _ _ ____ _____________________________________________ ________ ____
___ _ _ ____ _____________________________________________ ________ ____
___ _ _ ____ _____________________________________________ ________ ____
___ _ _ ____ _____________________________________________ ________ ____
___ _ _ ____ _____________________________________________ ________ ____
___ _ _ ____ _____________________________________________ ________ ____
___ _ _ ____ _____________________________________________ ________ ____
___ _ _ ____ _____________________________________________ ________ ____
___ _ _ ____ _____________________________________________ ________ ____
___ _ _ ____ _____________________________________________ ________ ____
___ _ _ ____ _____________________________________________ ________ ____
```

NU	LE	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	3		Line number (NUMERIC) IDENTIFIER DATA ELEMENT INVALID ABSENCE
2	1		Embedded level (ALPHABETIC)
			Its value must be numeric and included between 1 and 9. Value '1' must be unique in an Extraction Master Path because it corresponds to the entry point.
			Input in this field is relevant in S-, A-, and V- type lines.
3	1		Line type (ALPHABETIC) INVALID ABSENCE UPPERCASE
		S	Sequencing
		A	Access
		I	Input conditions
		O	Output conditions
		P	Presentation
		V	Virtual cursor
		*	Comments (printed in the validation report)
		blanc	Ignored by the validation procedure
			For more information, refer to Chapter 'Extraction Master Path Definition/Description', in this manual.
4	4		Code of reference cursor (ALPHABETIC) UPPERCASE
			Required on S-, A-, and V-type lines.

NU	LE	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			On S- and A-type lines, it specifies the cursor code associated with the PAF table. It allows for the extraction sequencing and identifies the occurrence extracted when the identification criterion is created. For more information, refer to Chapter 'The Extraction Master Path', Subchapters 'The Extraction Sequence (S-type lines) and (A-type lines)'.
			On V-type lines, it defines a virtual cursor code for the selective extraction of an occurrence. For more information, refer to Chapter 'The Extraction Master Path', Subchapter 'Selective Extraction (V-type lines)'.
5	40		Selection or comment (ALPHABETIC)
			Required on S-, A-, P-, I-, and O-type lines. For more information, refer to Chapter 'The Extraction Master Path'.
6	8		Virtual & presentation cursor (ALPHABETIC)
			The first four characters are a virtual cursor and the last four are a presentation cursor. For more information, refer to Chapter 'The Extraction Master Path', Subchapters 'Selective Extraction (V-type line)' for the virtual cursor and 'Presentation (P-type line)' for the presentation cursor.
7	4		Processing options (ALPHABETIC) UPPERCASE
			'O' column: Print option (for S-type lines)
		blank	Storing of the occurrence in the extraction result file; this data can be edited.
		N	Storing of the occurrence in the temporary Workfile; this data cannot be edited.
			'P' column: Definition of en entry point
		P	Definition of an occurrence as an entry point.
			'D' column: Delimiter value
		blank	Default value in the description syntax of the extraction on S- and A-type lines.
		,	Default value on P-type lines. If these default values are not convenient, the user may specify other values in this field. Warning: The "-" and "'" values are not authorized.
			'V' column: Value of the delimiter which bounds a constant on a P-type line (see Chapter 'The Extraction Master Path', Subchapter 'Presentation (P-type line).
		}	Default value If this value is not convenient, the user may modify it in this field. Warning: The "-" and "'" values are not authorized.

Chapter 9. PTEx: the Extraction Master Path

Extraction Sequence (S-type lines)

Data is extracted from the Database according to a path -- a sequencing -- based on the cross-references existing between entities in the metamodel.

This is why a thorough knowledge of the metamodel being used is essential.

The metamodel can be the VisualAge Pacbase classic metamodel or the VA Pac WorkStation metamodel for the Pacdesign or Pacbench modules (which varies with the methodology in use).

An Extraction Master Path can be thought of as a tree whose branches subdivide and explore the Specifications Dictionary in finer and finer detail.

The syntax used to describe the sequencing of an extraction is similar to the language used for online navigation within the Database.

In other words, you should ask yourself the following questions before anything else:

1. What data do I need to extract ?
2. What online input is necessary to access the screens which correspond to the extracted data?

Extraction sequencing is entered on S-type lines in the SELECTION-COMMENT field of the Y7E.....D screen.

An extraction sequence is composed of three elements; the entity type, the occurrence, and the type of line.

S-type lines examples are given at the end of this subchapter and two complete PTEx examples are presented at the end of this manual.

1. ENTITY TYPE:

The entity type to be entered corresponds to the entity type that is entered in the online CHOICE field.

NOTE: If the entity is a WorkStation entity, its type must be coded according to the following format:

//ü__CCC

where 'ü' is the Methodology code ('M' for Merise, 'D' for YSM...),

and 'CCC' is the entity local code.

To know the value of these codes, used the PCM command in the GENERATION AND PRINT COMMANDS screen (CHOICE: GP).

The code of the methodology is entered on one character in the ENTITY field.

The list of values for the various methodologies is the following:

```
M for Merise
D for YSM
A for SSADM
O for OMT
F for IFW
```

1. OCCURRENCE:

The occurrence must be separated from the entity type (default value: SPACE, modifiable in column 'D'). It is identified by the following two elements separated by a dash (-):

1. Cursor code identifying the hierarchically higher PAF Table. The extraction sequence depends exclusively on the cursor code which must be unique in an Extraction Master Path.
2. Contents of the occurrence code: PAF Column code.

   An index of the PAF column codes is available at the Support internet site at the following address:

   www-1.ibm.com/support/docview.wss?rs=37&uid=swg27005477

   (follow the 'PAF module' link).

   NOTE: If the occurrence is an occurrence of a WorkStation entity, the code of the Column code is the code of the Data Element called by the User Entity which supports this entity in the Database.

   In this case the cursor code and the Data Element/Column code are separated by two dashes (--). See Example e) at the end of this subchapter.

1. TYPE OF LINE - TABLE TO EXTRACT:

This type of line corresponds to the coding in the second part of the online CHOICE field (ex: //M DOM DOM-COEU X1MCD, where X1MCD is the type of line). It completes the entity type to specify which table is to be extracted.

This coding must be separated from the occurrence identifier (default value: SPACE, modifiable in column 'D').

The list of the table codes to be extracted for entities managed by the WorkStation is available on the Support internet site , at the following address:

www-1.ibm.com/support/docview.wss?rs=37&uid=swg27005477

EXAMPLES

```
LIN : L T REF. SELECTION-COMMENT
010 : 1 S PGM  P
020 : 2 S LDST P PGM-CPGM CD
```

On line 010, the L column (which corresponds to the embedded level) does not need to be filled in since this line correponds to the initial entry.

On line 020, the list of Data Structures for each occurrence of the Program entity will be requested.

```
LIN : L T REF. SELECTION-COMMENT
010 : 1 S SEG  S
020 : 2 S LDEL S SEG-CSEG CE
```

The Data Elements (LDEL cursor) belonging to each occurrence of the Segment (SEG cursor) entity will be listed.

```
LIN : L T REF. SELECTION-COMMENT
010 : 1 S PGM  P
020 : 2 S PGMP P PGM-CPGM P
```

The '-P' lines of each occurrence of the Program entity will be listed.

```
LIN : L T REF. SELECTION-COMMENT
010 : 1 S DOM  //M DOM
020 : 2 S LMCD //M DOM-CUEO X1MCD
```

The Conceptual Data Models for each occurrence of the Domain entity will be listed.

```
LIN : L T REF. SELECTION-COMMENT
010 : 1 S PHA  //M PHA
020 : 2 S FLC  //M CHA PHA--PHACH
```

The definition of the Flowchart associated with each occurrence of the Phase entity will be listed.

Explanation of this coding:

On the Definition of a Phase appears the code of its associated Flowchart. The Data Element which contains the occurrence code of the associated Flowchart must then be extracted.

For the fourth field - FLOWCHART - the Data Element code (.PHACH) corresponding to the VALUE column will be read. It will be coded -PHACH in the Extraction Master Path, separated from the cursor code by the second dash (-).

More generally, all Data Elements which describe a User Entity AND whose code is prefixed by a dot, must correspond to a Column code prefixed by a dash instead of the dot.

NOTE: At the end of this manual, an appendix presents two examples of Extraction Master Paths. One uses the WorkStation metamodel, and the other one uses the classic VisualAge Pacbase metamodel.

Extraction Sequence (Particular Cases)

Ambiguities may appear when selecting certain tables for extraction, particularly cross-reference tables. In this case the occurrence identifier and the table must be added.

EXAMPLE: The objective is to list the uses of a Segment in Programs.

These uses may be viewed on the 'S....XP.....CP..' and the 'S....XP.....W.....' screens.

The first screen lists the Segments called by the Data Structures that are called in the Program (P.....CD). The second screen lists the Segments called in the Working Storage Section of the Program (P.....W).

This is why the following may be ambiguous:

```
LIN : L T REF. SELECTION-COMMENT
020 : 2 S SEGP S SEG-CSEG XP
```

Therefore, it should be written as:

```
        S SEG-CSEG XP PGM-CPGM CD
and/or  S SEG-CSEG XP PGM-CPGM W
```

At this step in the pathway, if the PGM cursor has not been defined, i.e. if no extraction has been requested on the Program entity, the following lines must be entered:

```
        S SEG-CSEG XP * CD
and/or  S SEG-CSEG XP * W
```

The result will be the uses of the Segment in the -CD and/or -W of all the Programs of the queried Library.

Extraction Sequence (A-type lines)

Like an S-type line, an A-type line expresses an extraction selection. To condition an extraction by testing the value of a Data Element (which belongs to the cursor, BUT which is not an identifier) the user must write an A-type line and enter the hierarchially superior cursor in the 'VIR.' field.

EXAMPLE:

The objective is to find and list the uses of the Data Elements in the '-CD' of Programs.

```
LIN : L T REF. SELECTION-COMMENT              VIR.PRES
010 : 1 S DEL  E
020 : 2 A PCDE P * CD                         DEL
030 :   O      CDEL = DEL-CDEL
040 :   O      OR CRES = DEL-CDEL
050 :   S PGM  P PCDE-CPGM
```

NOTE: In this example the identifiers of the PCDE cursor are P (for Program) and CD (for Data Structure); CDEL is an additional Data Element of the table.

First, extract all the Data Elements, then select the -CD lines of Programs calling the Data Elements (the Programs not having been called in a preceding extraction).

The pathway does not logically follow. The link between the Data Elements and the Programs is assured by an A-type line which contains the cursor of the Data Element Table in the first four characters of the VIR.PRES field. This type of line includes a selection expressed with an asterisk.

Once the objects of the extraction are identified, the extraction conditions (the "filter") are entered on one or several O-type lines which must refer to the cursor specified in the VIR.PRES field of the A-type line.

NOTE: The filter is discussed in the next subchapter.

Conditions and Filters (I and O-type lines)

INTRODUCTION

The condition line authorizes the extraction from a Table (Input).

The filter selects the occurrences resulting from the extraction (Output).

CONDITIONS : I-TYPE LINES

Conditioning an extraction is particularly useful when the extraction must follow a search path starting from the data which fulfills the condition(s). Data occurrences that are excluded by the condition will not be extracted.

The condition is expressed on an I-type line, in COBOL, in the form of a Boolean operator + the expression. The condition references a hierarchically greater Table already extracted.

EXAMPLE: The objective is to list the occurrences used by a Program.

For the Report entity, occurrences are to be extracted only if the call of their Data Structures in the -CD of the Program is an I- or J-type line.

```
LIN : L T REF. SELECTION-COMMENT
010 : 1 S PGM  P
020 : 2 S LDST P PGM-CPGM CD
030 : 3 S LRPT D LDST-CDST LR
040 :   I      LDST-ODSTUS = 'I' OR 'J'
```

FILTERS: O-TYPE LINES

Filters make it possible to screen out unwanted occurrences of an extraction. A filter is expressed on an O-type line in SQL-PAF language using the WHERE clause of the EXEC PAF DECLARE command.

The syntax of this language is described in chapter 'Implementation in User Programs', subchapter 'Syntax of the SQL-PAF Language', paragraph 'Cursor Declaration'.

The Filter uses a column of the current Table as a filter criterion.

NOTE: At least one filter line is mandatory after an

A-type selection line.

EXAMPLE: The objective is to list the texts called in a Volume whose occurrence code is RAP001 and to list the Data Elements linked to these texts.

```
LIN : L T REF. SELECTION-COMMENT
010 : 1 S TXT  T
020 : 2 S XVOL T TXT-CTXT XV
030 :   O      CVOL = 'VOL001'
040 : 3 S TXTD T XVOL-CTXT D
050 :   O      CDEL <> SPACE
```

Selective Extraction (V-type line)

The processing loops of an extraction may cause the same occurrence to be extracted several times. Such might be the case if a Data Element is used several times in a Program. The repeated extraction of these occurrences may or may not be relevant. If it is not, the principle of Selective Extraction is used.

A Selective Extraction associates each occurrence with an identification criterion which differs from the one associated with the extraction level in the PTEx.

EXAMPLE: The objective is to list the uses of the Data Elements in Programs. Each occurrence will be identified by the following criteria, according to the number of levels in the extraction:

```
1  Program         (PPPPPP) => PGM PPPPPP
2  Data Structures (DS)     => PGM PPPPPP LDST DS
3  Segment         (DSSS)   => PGM PPPPPP LDST DS LSEG DSSS
4  Data Element    (EEEEEE) => PGM PPPPPP LDST SD LSEG DSSS
                               LDEL EEEEEE
```

To link the Data Elements directly to the Program which uses them, a virtual or selection cursor code (DELP) must be defined. This cursor will have an embedded level of 2 since the path passes directly from the Program to the Data Elements:

```
                          => PGM PPPPPP DELP EEEEEE
```

VARIANT: To link the Data Elements to their Data Structures, the selection cursor code is DELD. This cursor will have an embedded level of 3 (Program, Data Structures, Data Elements).

```
                          => PGM PPPPPP LDST DS DELD EEEEEE
```

The selection cursor code is positioned in the Selection- Comment field (on S, I, or O-type line) of the entity to be selected, in the first four characters of the VIR.PRES field. Then, a V-type line is added at the end of the PTEx. On this line, the embedded level of the selection cursor is entered in the L column and the selection cursor code itself is entered in the REF. column.

```
LIN : L T REF. SELECTION-COMMENT            VIR.PRES
010 : 1 S PGM  P
020 : 2 S LDST P PGM-CPGM CD
```

```
030 : 3 S LSEG D LDST-CDST LS
040 :   I      (LDST-ODSTUS NOT = 'I' AND 'J'
050 :   I      AND LDSC-ODSTOR = 'V' OR 'S')
060 : 4 S LDEL S LSEG-CSEG CE                DELP
070 :   O      CDEL <> FILLER
900 : 2 V DELP

VARIANT :

LIN : L T REF. SELECTION-COMMENT             VIR.PRES
... : . . ...  ......................
030 : 3 S LSEG D LDST-CDST LS
... : . . ..........................
060 : 4 S LDEL S LSEG-CSEG CE                DELD
070 : . . ..........................
900 : 2 V DELD
```

A selection cursor must be entered as many times as branches of the extraction tree lead to the same entity starting from the same reference entity.

For example, different extraction paths may finally lead to the Data Elements used in a Program. In the preceding example, the DELP cursor should be used several times, each time referenced to the same V-type line.

NOTE: The SELECTION-COMMENT field of a V-type line can contain a title or a label for documentation purposes.

Presentation (P-type line)

Using presentation lines (or format lines), the user may:

- Select columns which describe occurrences resulting from an extraction,
- Specify a particular presentation for these selections.

These lines may be used to describe User Extractors (E-type PTEx) and Macro-Commands (M-type PTEx).

P-type lines are not required. Whether P-type lines are entered or not, a PTEx always creates an unformatted result file which contains all the Tables extracted (except those withheld by the user; see the 'Print Option', in the 'Processing Options' of the 'Extraction Master Path: Definition / Description' chapter).

USES OF P-TYPE LINES

In User Extraction Programs, Presentation lines may be used to:

- 'Draw' personalized list layouts,
- Automatically format user input for batch procedures such as GPRT or EXTR (but this is useless for UPDP).

In Macro-Commands, Presentation lines allow the user to specify a format which will be taken into account in a PDM+ Volume. Such a format is called by a G-type line in a Master Outline.

>>>: For more details, refer to the Personalized Documentation Manager Reference Manual.

SYNTAX OF P-TYPE LINES (SELECTION-COMMENT field)

P-type lines are written in positional language, based on the juxtaposition of the following parameters:

PPP,LLL,content

PPP: Positioning:

Numeric, 3 characters maximum, indicates the position of the beginning of the transfer into the reception field.

LLL: Length:

Numeric, 3 characters maximum, indicates the length of the field to transfer into the reception field.

content:

Field to transfer:

- Constant (bounded by the value present in column 'V' of the OPDV field, '}' by default).
- Table column (column code) extracted by the current cursor (generally a column which contains an occurrence code).
- Table column (cursor-column) extracted by another cursor.

  A column from another table may be called in 'content'; so columns coming from different tables may be formatted the same way.

'PPP,LLL,content' can be repeated as many times as necessary; knowing that the expanded length of the reception field must not exceed the value specified by the MAX. RECORD SIZE field in the PTEx Definition.

NOTE: A presentation may be written on several consecutive P-type lines provided that the presentation cursor is entered on the first line only. See below Paragraph 'Positioning Presentation Cursors' for more details.

Several presentations for an extracted occurrence may be defined, and one or several of them may be selected. For example, an E-type occurrence corresponds both to a Data Element and to a Property. The user may then define a specific presentation for a Data Element, and another presentation for a Property.

A P-type line may be conditionned if it is directly followed by an I-type line.

```
Ex:  P  PRE1  1,12,'DATA ELEMENT',14,6,CDEL
     I         DEL-TDEL = 'R'
     P  PRE2  1,8,'PROPERTY',10,6,CDEL
     I         DEL-TDEL = 'I'
```

If a presentation is associated with several cursors, the cursor code must be replaced by an asterisk '*'.

```
Ex:  P  PRE1  1,12,'DATA ELEMENT',14,6,CDEL
     I         *-TDEL = 'R'
```

The comma is the default value for delimiting the parameters of P-type lines. Another character can be used as a delimiter if it is specified in column 'D' of the OPDV field of each P-type line concerned.

SPECIFIC SYNTAX OF THE MACRO-COMMAND P-TYPE LINES

The SELECTION-COMMENT field in the description of a User Entity occurrence recognizes the '$VF' PDM presentation parameter which corresponds to the character used for vertical separators. Presentation is limited to 132 characters (to avoid truncations at the time of printing).

All or part of a cursor presentation may be taken into account in a PDM+ Volume. For more information refer to the description of a Master Outline in the 'Personalized Documentation Manager' Reference Manual.

POSITIONING PRESENTATION CURSORS

A presentation cursor is positioned inside the selection (S, A, I, or O-type lines) of the table to be formatted in the last four characters of the VIR.PRES field.

NOTE: If a V-type line exists for the cursor that is to be formatted, it is recommended that the cursor be positioned directly on this line (see line 900 of variant 2 shown below).

The P-type line is entered, either:

- directly in the selection of the Table to be formatted, in which case entering the presentation cursor in the REF. field is not necessary (see variant 2), or
- at the end of the Extraction Master Path as in variant 1.

The REF. field must contain the presentation cursor entered previously, except in the case of variant 2.

You may enter as many P-type lines as there are presentations to be extracted. A presentation may also be described on several P-type lines (see the NOTE above).

Conversely, a specific presentation may be associated with the same cursor.

EXAMPLE:

```
LIN : L T REF. SELECTION-COMMENT             VIR.PRES
010 : 1 S PGM  P
020 : 2 S LDST P PGM-CPGM CD
030 : 3 S LSEG D LDST-CDST LS
040 :   I      (LDST-ODSTUS NOT = 'I' AND 'J'
050 :   I      AND LDST-ODSTOR = 'V' OR 'S')
060 : 4 S LDEL S LSEG-CSEG CE
070 :   O      CDEL <> FILLER
080 :   P      2,7,}W1EX UE},9,6,CDEL
```

VARIANT 1:

```
LIN : L T REF. SELECTION-COMMENT             VIR.PRES
... : . . .... ..............................
060 : 4 S LDEL S LSEG-CSEG CE                PDEL
070 :   O        CDEL <> FILLER
... : . . .... ..............................
999 :   P PDEL 2,7,}W1EX UE},9,6,CDEL
```

VARIANT 2:

```
LIN : L T REF. SELECTION-COMMENT             VIR.PRES
... : . . ...  ..............................
060 : 4 S LDEL S LSEG-CSEG CE                DELP
... : . . ....................................
900 : 2 V DELP SELECTION & PRES. SUPPORT         PDEL
910 :   P      2,7,}W1EX UE},9,6,CDEL
```

```
Above, the extraction finds the Data Element lines
which are then ready to be processed by EXTR, the
batch extraction procedure.
```

Chapter 10. PAF+ Implementation

User Input: E-Type PTEx

EXECUTION OF A USER EXTRACTOR: USER INPUT

- Several *-type lines, at least one is required.

POS.	LEN.	VALUE	MEANING
2	1	*	Line type
3	8	uuuuuuuu	User code
11	8	pppppppp	Password
19	3	bbb	Library code
22	4	ssss	Session number. Default value: current session
26	1		Session status if frozen:
		' '	Frozen
		T	Test
27	1		Library Network Option:
		C	Default value: Selected and upper libraries See the Character Mode User Interface Guide or the Developer workbench online help for other possible values

- Required 'X'- and 'Y'- type lines. One of their purposes is to qualify the extraction's scope.

Several lines may be entered so as to parameterize different executions of one User Extractor program.

POS.	LEN.	VALUE	MEANING
2	1	X	Line type
3	4		Cursor code. Default value: 1rst cursor
7	32		Occurrence code.
			Default value: All occurrences. The '*' generic character is authorized.
39	1	1	Printing of a debug report. Default value : no debug report
40	6		Number of records in SYSPAF file. Equivalent to PAF/SIZE parameter. Default value : 10

- A 'Y' line

POS.	LEN.	VALUE	MEANING
2	1	Y	Line type
3	32		Starting bound
35	32		Ending bound

For technical information, please refer to the 'Developer's Procedures' Manual, chapter 'Personalized Extractions / Automated Documentation', subchapter 'Personalized Extractions - XPAF', section 'XPAF: Validation of the Extraction Master Path'.

PTEx examples - Extraction Master Path Validation

The following pages contain two examples of Extraction Master Path validation reports (XPAF Procedure).

An XPAF execution report is composed of:

1. The "COMMENT OR ERROR" page;
2. The user input,
3. The EXTRACTION MASTER PATH DESCRIPTION LINES;
4. The extraction simulation.

The first Extraction Master Path example explores the Data Element entity cross-references in the Specifications Database metamodel.

The second example explores the Flowchart entity cross-references specific to the VA Pac WorkStation's metamodel.

Example of a PAF+ Extraction User Program/CICS

```
000100 //*****************************************************************
000120 //* VISUALAGE PACBASE
000130 //*
001100 //*              - JCL EXAMPLE -
001200 //*    USER EXTRACTOR SUBMISSION JCL
001300 //*    ADD A '*' LINE    (USER, PASSWORD, LIBRARY)
001400 //*    ****  ADD A X-TYPE LINE (REFER TO THE PAF USER REF.)
001500 //*              COL. 2    :   X
001600 //*              COL. 3    :   PAF CURSOR
001700 //*              COL. 7    :   UEO CODE
001800 //*              COL. 39   :   DEBUG TYPE (0 OU 1)
001900 //*              COL. 40   :   NUMBER OF PAF BUFFERS
002000 //*    ****  ADD A Y-TYPE LINE
002100 //*              COL. 3    :   START LIMIT
002200 //*              COL. 35   :   END LIMIT
002210 //*    CHANGE DCB OF PAC7SQ IN RELATION WITH THE USER ENTITY
```

```
002220 //*
002300 //*****************************************************************
002400 //PTEXJCL PROC BASE=$BASE,                CODE OF VAPAC DATABASE
002600 //      INDSV='$INDSV',                INDEX OF VSAM SYSTEM FILES
002700 //      INDSN='$INDSN',            INDEX OF NON-VSAM SYSTEM FILES
002800 //      INDUV='$INDUV',                  INDEX OF VSAM USER FILES
002900 //*:    VSAMCAT='$VCAT',                       USER VSAM CATALOG
003000 //*:    SYSTCAT='$SCAT',                     SYSTEM VSAM CATALOG
003100 //      STEPLIB=,                   USER LIBRARY OF LOAD-MODULES
003200 //      SORTLIB=,                                SORT LIBRARY
003300 //      OUT=$OUT,                                OUTPUT CLASS
003400 //      UWK=$UWK,                                   WORK UNIT
003500 //      SPAMB='(TRK,(1,1),RLSE)',             REQUESTS FILE SPACE
003600 //      SPASQ='(TRK,(5,1),RLSE)',               RESULT FILE SPACE
003700 //      SPAWK='(TRK,(5,1),RLSE)',                  WORK FILE SPACE
003800 //      CYL='(3,1)'                                 SORTWORK SPACE
005000 //*****************************************************************
005100 //INPUT  EXEC PGM=BVPTU001
005200 //**********************
005300 //STEPLIB  DD DSN=&STEPLIB,DISP=SHR
005302 //         DD DSN=$BCOB,DISP=SHR
005400 //CARTE    DD DDNAME=SYSIN
005500 //PAC7MB   DD DSN=&&XPAFMB,DISP=(,PASS),
005600 //          UNIT=&UWK,
005700 //          SPACE=&SPAMB,DCB=BLKSIZE=3440
005800 //MAXKEY  EXEC PGM=IDCAMS
005900 //**********************
006000 //*:STEPCAT DD DSN=&VSAMCAT,DISP=SHR
006100 //SYSPRINT DD SYSOUT=&OUT
006200 //SYSPAF   DD DSN=&&SYSPAF,DISP=(NEW,KEEP),
006300 //         SPACE=(CYL,(3,3)),
006400 //         LRECL=1031,RECORG=KS,KEYOFF=0,KEYLEN=12
006500 //MAXKEY   DD DSN=&INDSN..BVPSY(MAXKEY),DISP=SHR
006600 //SYSIN    DD DSN=&INDSN..BVPSY(REPRO999),DISP=SHR
006700 //*
006800 //-------  EXEC PGM=------
006900 //**********************
007000 //STEPLIB  DD DSN=&STEPLIB,DISP=SHR
007002 //         DD DSN=$BCOB,DISP=SHR
007100 //SORTLIB  DD DSN=&SORTLIB,DISP=SHR
007200 //*:STEPCAT  DD DSN=&VSAMCAT,DISP=SHR
007300 //PAC7AN   DD DSN=&INDUV..&BASE.AN,DISP=SHR
007400 //PAC7AR   DD DSN=&INDUV..&BASE.AR,DISP=SHR
007402 //PACGGN   DD DSN=&INDSV.BVPGN,DISP=SHRP=SHR
007404 //PACGGR   DD DSN=&INDSV.BVPGR,DISP=SHRP=SHR
007406 //PACGGU   DD DSN=&INDSV.BVPGU,DISP=SHRP=SHR
007500 //PAC7AE   DD DSN=&INDSV..BVPAE,DISP=SHR
007600 //SYSPAF   DD DSN=&&SYSPAF,DISP=(OLD,KEEP)
007700 //PAC7MB   DD DSN=&&XPAFMB,DISP=(OLD,DELETE)
007800 //PAC7SO   DD SPACE=&SPAWK,DCB=BLKSIZE=13080,
007900 //         UNIT=&UWK
008000 //PAC7SQ   DD DSN=&&PAC7SQ,DISP=(,PASS),
008100 //         UNIT=&UWK,
008200 //         DCB=(RECFM=FB,LRECL=80,BLKSIZE=6080),
008300 //         SPACE=&SPASQ
```

```
008400 //PAC7DB   DD SYSOUT=&OUT
008500 //SORTWK01 DD UNIT=&UWK,SPACE=(CYL,&CYL,,CONTIG)
008600 //SORTWK02 DD UNIT=&UWK,SPACE=(CYL,&CYL,,CONTIG)
008700 //SORTWK03 DD UNIT=&UWK,SPACE=(CYL,&CYL,,CONTIG)
008800 //SYSOUT   DD SYSOUT=&OUT
008900 //SYSUDUMP DD SYSOUT=&OUT
009000 //SYSPRINT DD SYSOUT=&OUT
009100 //         PEND
010000 //PTEXJCL EXEC PTEXJCL
010200 //
```

Example of a PAF+ Extraction User Program/IMS

```
000100 //*************************************************************
000120 //* VISUALAGE PACBASE
000130 //*
001000 //* INSTALLATION  - JCL EXAMPLE -
001100 //*    USER EXTRACTOR SUBMISSION JCL
001200 //*    IN INPUT, ADD A '*' LINE    (USER, PASSWORD, LIBRARY)
001300 //*    ********  ADD A X-TYPE LINE (REFER TO THE PAF USER REF. )
001400 //*               COL. 2    :   X
001500 //*               COL. 3    :   PAF CURSOR
001600 //*               COL. 7    :   UEO CODE
001610 //*               COL. 39   :   DEBUG TYPE (0 OU 1)
001620 //*               COL. 40   :   NUMBER OF PAF BUFFERS
001650 //*    ********  ADD A Y-TYPE LINE
001700 //*               COL. 3    :   START LIMIT
001800 //*               COL. 35   :   END LIMIT
002100 //*    CHANGE DCB OF PAC7SQ IN RELATION WITH THE USER ENTITY
002200 //*************************************************************
002300 //PTEXJCL PROC BASE=$BASE,                CODE OF VAPAC DATABASE
002400 //      INDSV='$INDSV',               INDEX OF VSAM SYSTEM FILES
002500 //      INDSN='$INDSN',            INDEX OF NON-VSAM SYSTEM FILES
002600 //      INDUV='$INDUV',                  INDEX OF VSAM USER FILES
002700 //*:    VSAMCAT='$VCAT',                       USER VSAM CATALOG
002800 //*:    SYSTCAT='$SCAT',                    SYSTEM VSAM CATALOG
002900 //      STEPLIB=,                    USER LIBRARY OF LOAD-MODULES
003000 //      SORTLIB=,                                 SORT LIBRARY
003100 //      OUT=$OUT,                                 OUTPUT CLASS
003200 //      UWK=$UWK,                                    WORK UNIT
003300 //      SPAMB='(TRK,(1,1),RLSE)',             REQUESTS FILE SPACE
003400 //      SPASQ='(TRK,(5,1),RLSE)',               RESULT FILE SPACE
003500 //      SPAWK='(TRK,(5,1),RLSE)',                  WORK FILE SPACE
003600 //      CYL='(3,1)',                                SORTWORK SPACE
003700 //          PSBLIB='$PSBLIB',    IMS PSBLIB
003800 //          DBDLIB='$DBDLIB',    IMS DBDLIB
003900 //          RESLIB='$RESLIB',    IMS RESLIB
004000 //          PROCLIB='$PRCLIB',   IMS WORKLIB
004100 //         BUF=40,SPIE=0,TEST=0,EXCPVR=0,RST=0,PRLD=,SRCH=0,
004200 //         CKPTID=,MON=N,LOGA=0,FMTO=T,DBRC=$DBRC,IRLM=$IRLM
004300 //*************************************************************
004400 //INPUT  EXEC PGM=BVPTU001
004500 //**********************
004600 //STEPLIB  DD DSN=&STEPLIB,DISP=SHR
004700 //         DD DSN=$BCOB,DISP=SHR
004800 //CARTE    DD DDNAME=SYSIN
```

```
004900 //PAC7MB   DD DSN=&&XPAFMB,DISP=(,PASS),
005000 //          UNIT=&UWK,
005100 //          SPACE=&SPAMB,DCB=BLKSIZE=3440
005200 //*
005300 //______  EXEC PGM=DFSRRC00,REGION=$REGSIZ,
005400 //        PARM=(DLI,______,______,&BUF,
005500 //        &SPIE&TEST&EXCPVR&RST,&PRLD,
005600 //        &SRCH,&CKPTID,&MON,&LOGA,&FMTO,,,&DBRC,&IRLM)
005700 //STEPLIB  DD DSN=&RESLIB,DISP=SHR
005800 //         DD DSN=&STEPLIB,DISP=SHR
005900 //         DD DSN=$BCOB,DISP=SHR
006000 //DFSRESLB DD DSN=&RESLIB,DISP=SHR
006100 //IMS      DD DSN=&PSBLIB,DISP=SHR
006200 //         DD DSN=&DBDLIB,DISP=SHR
006300 //*:STEPCAT DD DSN=&SYSTCAT,DISP=SHR
006400 //*:       DD DSN=&VSAMCAT,DISP=SHR
006500 //SYSOUT   DD SYSOUT=&OUT
006600 //SYSOUX   DD SYSOUT=&OUT
006700 //DDSNAP   DD SYSOUT=&OUT
006800 //PROCLIB  DD DSN=&PROCLIB,DISP=SHR
006900 //IEFRDER  DD DUMMY,
007000 //            DCB=(RECFM=VB,BLKSIZE=1920,LRECL=1916,BUFNO=2)
007100 //SYSUDUMP DD SYSOUT=&OUT,DCB=(RECFM=FBA,LRECL=121,
007200 //            BLKSIZE=605),SPACE=(605,(500,500),RLSE,,ROUND)
007300 //IMSUDUMP DD SYSOUT=&OUT,DCB=(RECFM=FBA,LRECL=121,
007400 //            BLKSIZE=605),SPACE=(605,(500,500),RLSE,,ROUND)
007500 //IMSMON   DD DUMMY
007600 //DFSVSAMP DD DSN=&INDSN..BVPSY(DFSVSAM8),DISP=SHR
007700 //SORTLIB  DD DSN=&SORTLIB,DISP=SHR
007800 //SORTWK01 DD UNIT=&UWK,SPACE=(CYL,&CYL,,CONTIG)
007900 //SORTWK02 DD UNIT=&UWK,SPACE=(CYL,&CYL,,CONTIG)
008000 //SORTWK03 DD UNIT=&UWK,SPACE=(CYL,&CYL,,CONTIG)
008100 //BVP7AE   DD DSN=&INDSV..BVPAE,DISP=SHR
008200 //B7AN$BASE   DD DSN=&INDUV..&BASE.AN,DISP=SHR
008300 //B7AR$BASE   DD DSN=&INDUV..&BASE.AR,DISP=SHR
008310 //BVP7GN   DD DSN=&INDSV..BVPGN,DISP=SHR
008320 //BVP7GR   DD DSN=&INDSV..BVPGR,DISP=SHR
008330 //BVP7GU   DD DSN=&INDSV..BVPGU,DISP=SHR
008400 //B7PA$BASE   DD DSN=&INDUV..&BASE.PA,DISP=SHR
008500 //B7P1$BASE   DD DSN=&INDUV..&BASE.P1,DISP=SHR
008600 //PAC7MB   DD DSN=&&PAC7MB,DISP=(OLD,DELETE)
008700 //PAC7SO   DD UNIT=&UWK,SPACE=&SPAWK,DCB=BLKSIZE=27904
008800 //PAC7SQ   DD DSN=&&PAC7SQ,DISP=(,PASS),UNIT=&UWK,
008900 //            DCB=(RECFM=FB,LRECL=80,BLKSIZE=6080),
009000 //            SPACE=(&SPASQ,RLSE)
009100 //PAC7DB   DD SYSOUT=&OUT
009200 //*
```

Example of a PAF+ Extraction User Program/WNT

```
000010 REM * -------------------------------------------------------
000020 REM *      VISUALAGE PACBASE
000030 REM *
000040 REM * -------------------------------------------------------
000050 REM *         - EXAMPLE OF PAF+ PROCEDURE EXECUTION -
000060 REM *
```

```
000070 REM * -------------------------------------------------------
000080 REM *  INPUT FILE DESCRIPTION :
000090 REM *  ADD A '*' LINE    (USER, PASSWORD, LIBRARY)
000100 REM *  ****  ADD A X-TYPE LINE (REFER TO THE PAF USER REF.)
000110 REM *             COL. 2    :   X
000120 REM *             COL. 3    :   PAF CURSOR
000130 REM *             COL. 7    :   UEO CODE
000140 REM *             COL. 39   :   DEBUG TYPE (0 OU 1)
000150 REM *             COL. 40   :   NUMBER OF PAF BUFFERS
000160 REM *  ****  ADD A Y-TYPE LINE
000170 REM *             COL. 3    :   START LIMIT
000180 REM *             COL. 35   :   END LIMIT
000190 REM *  CHANGE DCB OF PAC7SQ IN RELATION WITH THE USER ENTITY
000200 REM *
000210 REM *
000800
000900 Set WshShell = Wscript.CreateObject("Wscript.Shell")
001000 Set Args = Wscript.Arguments
001100 Set WshEnv = WshShell.Environment("PROCESS")
001200 Set WshVolEnv = WshShell.Environment("VOLATILE")
001300
001400 Dim FSO
001500 Set  FSO = CreateObject("Scripting.FileSystemObject")
001600 Dim ObjProcess
001700 Set  ObjProcess = CreateObject("BvpWsh.Process")
001800
001900 Rep_BVP="HKLM\SOFTWARE\IBM\BVP VisualAge Pacbase\Server\BVP"
002000
002100 Rep_Proc = WshShell.RegRead(Rep_BVP & "_SYS\PROC\")
002200 Rep_NLS = WshShell.RegRead (Rep_BVP & "_SYS\NLS\")
002300 Rep_SKEL = WshShell.RegRead (Rep_BVP & "_SYS\SKEL\")
002400 Rep_PGM = WshShell.RegRead (Rep_BVP & "_SYS\PGM\")
002500
002600   WshEnv("PATH") = Rep_PGM & ";" & WshEnv("PATH")
002700
002800 ' Set COBOL environments :
002900 ' ------------------------
003000 Var = "HKLM\SOFTWARE\Micro Focus\NetExpress\"
003100
003200 On Error Resume Next
003300 NetExpress_Version = WshShell.RegRead (Var & _
003400                         "DefaultVersion")
003500 If Err.Number = 0 Then
003600    NetExpress = 1
003700 Else
003800    NetExpress = 0
003900 End if
004000
004100 On Error Resume Next
004200 NetExpress_Version = WshShell.RegRead (Var & "Version")
004300 If Err.Number = 0 Then
004400    ApplicationServer = 1
004500 Else
004600    ApplicationServer = 0
004700 End if
```

```
004800
004900
005000 If NetExpress = 1 or ApplicationServer = 1 Then
005100    Var = Var & NetExpress_Version & "\COBOL\"
005200    COBOL_Version = WshShell.RegRead (Var & "Version")
005300    Var = Var & COBOL_Version & "\Environment\"
005400    COBPATH =  WshShell.RegRead (Var & "PATH")
005500
005600    WshEnv("COBDIR") = WshShell.RegRead (Var & "COBDIR")
005700    WshEnv("COBHNF") = WshShell.RegRead (Var & "COBHNF")
005800    WshEnv("PATH") = COBPATH & ";" & WshEnv("PATH")
005900 Else
006000  wscript.echo "NetExpress or Application Server
006100                              not implemented. "
006200 End if
006300 '============================================================
006400
006500 ' EXECUTION of PAFPGP
006600 '-----------------------------------
006700
006800 ' 1-Delete PAF File
006900 If  FSO.FileExists(Rep_TMP & "\WPAF.tmp") Then
007000 Set FileD = FSO.GetFile(Rep_TMP & "\WPAF.tmp")
007100 FileD.Delete
007200 End If
007300
007400 ' 2-Assign database files ...
007500 WshEnv("PAC7AE") = Rep_SKEL & "\AE"
007600 WshEnv("PAC7AR") = Rep_BASE & "\AR"
007700 WshEnv("PAC7AN") = Rep_BASE & "\AN"
007800 WshEnv("PACGGN") = Rep_ABASE & "\AN"
007810 WshEnv("PACGGR") = Rep_ABASE & "\AR"
007820 WshEnv("PACGGU") = Rep_ABASE & "\GU"
007840
007900 ' 2bis- ... and  PAF File
007910 WshEnv("SYSPAF") = Rep_TMP & "\WPAF.tmp"
007920
008000 ' 2ter- Input file for PAF
008010 WshEnv("PAC7MB") = ...
008100
008200 ' 3-Assign of program  files
008300 ' ......................
008400
008500 ' 4- Execute ...
008600 Return = WshShell.Run("BVPAFPGP.exe ", 1, TRUE)
008700
008800 ' 5- If Error
008900 if err.number <> 0 and Return = 0 then
009000  wscript.echo "Syntax or RunTime Error"
009100 End if
009200
009300 if  Return <> 0 then
009400  wscript.echo "PAFPGP execution error"
009500 End if
```

```
009600
009700
009800 Wscript.Quit (Return)
009900
```

Example of a PAF+ Extraction User Program/UNIX

```
000100 #!/bin/sh
000200 #
000300 # EXEMPLE OF AN EXECUTION PROCEDURE USING A PAF+ PROGRAM
000400 # ------------------------------------------------------
000500 # (execution of PGPAFP program)
000600
000700 # Input file description :
000800 # - one '*' LINE    (USER, PASSWORD, LIBRARY)
000900 # - ONE X-TYPE LINE (REFER TO THE PAF USER REF. )
001000 #   COL. 2    :   X
001100 #   COL. 3    :   PAF CURSOR
001200 #   COL. 7    :   UEO CODE
001300 #   COL. 39   :   DEBUG TYPE (0 OU 1)
001400 #   COL. 40   :   NUMBER OF PAF BUFFERS
001500 # - one Y-TYPE LINE
001600 #   COL. 3    :   START LIMIT
001700 #   COL. 35   :   END LIMIT
001750
001800 # Cobol compiler environment variable :
001900 COBDIR=/usr/local/cobol
002000 export COBDIR
002100
002200 # PATH environment variable :
002300 PATH=.:$COBDIR/bin:$PATH
002400 export PATH
002500
002600 # LIBPATH environment variable :
002700 # (for AIX version)
002800 LIBPATH=/usr/lib:$COBDIR/lib
002900 export LIBPATH
003000
003100 # LD_LIBRARY_PATH environment variable :
003200 # (for HP-UX, OSF1, SunOS or Linux version)
003300 LD_LIBRARY_PATH=/usr/lib:$COBDIR/lib
003400 export LD_LIBRARY_PATH
003500
003600 # VisualAge Pacbase directory environment variable :
003700 PACDIR="/lv00/ll350/paclanx"
003800 export PACDIR
003900
004000 # COBPATH environment variable :
004100 # (current directory + VisualAge Pacbase programs directory)
004200 COBPATH=.:$PACDIR/system/gnt
004300 export COBPATH
004400
004500 # VisualAge Pacbase database name (mandatory) :
004600 BASE=BVAP
004700
004800 # VisualAge Pacbase environment variable (mandatory) :
```

```
004900 . $PACDIR/config/$BASE/PAC7AE.ini
005000 . $PACDIR/config/$BASE/PAC7AN.ini
005100 . $PACDIR/config/$BASE/PAC7AR.ini
005200 . $PACDIR/config/$BASE/PACGGN.ini
005300 . $PACDIR/config/$BASE/PACGGR.ini
005400 . $PACDIR/config/$BASE/PACGGU.ini
005500 SYSPAF=./wpaf
005600 export SYSPAF
005700 PAC7SO=./pafp.so
005800 export PAC7SO
005900 PAC7SQ=./pafp.sq
006000 export PAC7SQ
006100 PAC7DB=./pafp.db
006200 export PAC7DB
006300
006400 # Transaction input file assignment (mandatory) :
006500 PAC7MB=./MBPAFP
006600
006700 # User application environment variable :
006800 FILE1=./file1
006900 export FILE1
007000
007100 # Start execution :
007200 cobrun PGPAFP
007300
007400 # Deletion of the temporary files :
007500 if [ -r "$SYSPAF" ]
007600 then
007700    rm $SYSPAF*
007800 fi
```

Part Number: DDPAF000351A - 7517

Printed in USA