VisualAge Pacbase 2.5

VisualAge Pacbase WorkStation

Operations Manual

DSEXP000251A

1st Edition (August 1998)

This edition applies to the following licensed program:

- VisualAge Pacbase Version 2.5

Comments on publications (including document reference number) should be sent electronically through the Support Center Web site at:
http://www.software.ibm.com/ad/vapacbase/support.htm
or to the following postal address:
IBM Paris Laboratory
VisualAge Pacbase Support
30, rue du Château des Rentiers
75640 PARIS Cedex 13
FRANCE

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

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:

Intellectual Property and Licensing
International Business Machines Corporation
North Castle Drive, Armonk, New-York 10504-1785
USA

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 information which has been exchanged, should contact:

IBM Paris Laboratory
SMC Department
30, rue du Château des Rentiers
75640 PARIS Cedex 13
FRANCE

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.

Table Of Contents

Introduction

This manual describes the installation and operation of VisualAge Pacbase WorkStation on mainframe, OS/2, Windows NT or UNIX servers.

It is composed of chapters which describe:

- The installation procedure, the parameterization of graphical environments from which the WorkStation can be used and a description of the hard disk drive.
- The connection scripts that allow access to the Host Specifications Dictionary via the WorkStation.
- The start-up of the WorkStation, from specific graphical environments or from the system.
- The utilities that allow a quick starting-up and shut down of the WorkStation and that enable you to edit Volumes in RTF format.
- The script language that the user must use to create or modifiy connection scripts.

For up-to-date hardware and software configuration details, contact your help desk for complete information.

1 WorkStation Installation

1.1. WorkStation and Host

The WorkStation operates with VisualAge Pacbase, which can be installed on:

- a mainframe server;
- an OS/2, Windows NT or UNIX local server.

In order for the WorkStation to operate correctly, VisualAge Pacbase must be installed correctly on the chosen host.

To install VisualAge Pacbase onto your host, please refer to the *Operations Manual* specific to your platform (Vol. I - *Environment & Installation*; Chapter *Installation*).

When you use the WorkStation for the first time, make sure that the following functions have been installed correctly:

- Specifications Dictionary function;
- Personalized Documentation Manager extension;
- Customization.

1.2. Installation of the Host part

1.2.1. Function keys

For each VisualAge Pacbase Database installed on an OS/2, Windows NT or UNIX server that you want to access from the WorkStation, if function keys are not assigned or if they are assigned non-standard values, then you must either:

- Define the function keys according to the standard options,

or

- At least assign the PF7 key to inhibit implicit update.

For each Database, function keys are defined via the REST procedure. For complete information concerning this procedure, please refer to the *Operations Manual* (Vol. II - *Batch Procedures:* Administrator's Guide) specific to your platform.

1.2.2. Installation of a methodology

For each VisualAge Pacbase Database installed on an OS/2, Windows NT or UNIX server that you want to access from the WorkStation, you must install the chosen methodology in a dedicated library. "To install a methodology" means loading entities specific to this methodology into the Database, which will allow this methodology to be used.

These entities are pre-defined User Entities. Their definition and decription should *never* be modified after the methodology has been installed. To ensure this, make sure for example that WorkStation users have no update access to the library you installed the methodology in. Consequently, this library should be high enough in the library hierarchy to be above all the WorkStation user libraries, thus allowing to work with methodology entities in these libraries.

The host installation media contain transaction files related to methodologies. Each one contains all the entities specific to one methodology. To install a methodology on the host, you must run the UPDT procedure with the right file as input.

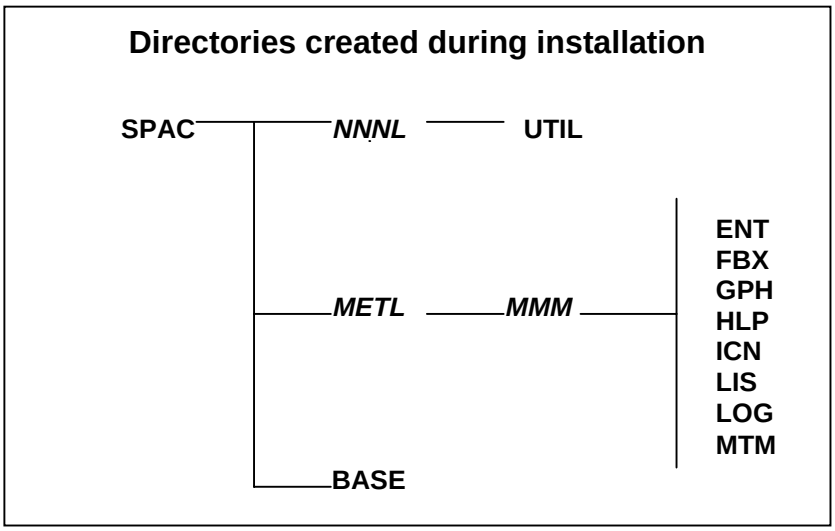
NOTE: For the user methodologies, the UPDT files are provided at installation in the directory of the methodology files: \SPAC\MET*L**MMM* (Refer to Subchapter *WorkStation Directory Structure* for more details about this directory).

This type of installation is documented in the VisualAge Pacbase Operations Manual specific to your platform (Vol. I - *Environment & Installation).*

You must also integrate the Methodology Choices in VisualAge Pacbase. These Choices are supplied at installation in the form of transactions files to be used as input in the PARM procedure (update of the user parameters).

For more details, refer to the *Operations Manual* specific to your platform (Vol. II - *Batch Procedures*: *Administrator's Guide*, Chapter "*Database Management Utilities*", Subchapter "*PARM: Update of User Parameters*").

1.3. WorkStation Directory Structure: Initial State and Evolution



In this diagram:

- *NNN* is the release number (e.g. 250);
- *L* is the language code (A for English, F for French);
- *MMM* is the methodology code (MER = MERISE, ADM = SSADM, DON = YSM, FAA = IFW, OMT)

The installation procedure creates the following directories:

- \SPAC is the root directory of the WorkStation. You can install it wherever you want to.
- \SPAC*NNNL*: this directory includes among others:
  - All executable files (.EXE) of the *NNNL* release.
  - The PACBASE.DAT file which contains all the WorkStation execution parameters.
  - The OK.VER file (the release installation flag).
  - The Communications Manager GSxxxxxx.PRM and GSxxxxxx.TAB parameter files (For complete information, refer to the *Communications Manager and Paclink Utility* Manual).
  - The CGIPAC.FON file which contains the font used by the Communications Manager in Trace mode (For complete information on the trace mode, refer to the *Communications Manager and Paclink Utility* Manual, Chapter "*Communications Manager*", Subchapter "*Description of the Communications Manager Window*").
- \SPAC*NNNL*\UTIL: this directory includes:
  - Utilities for starting up the WorkStation directly;
  - The NEWS*L*.TXT file (*L* for the methodology language), which can be accessed with any text editor, contains a text on the enhancements of the WorkStation installed release.
- \SPAC\MET*L**MMM* includes all parameters describing a specific methodology (*MMM*) in a specific language (*L*). There are 8 different types of these parameter files each one being located in its own subdirectory.
  - ENT One .ENT file for each managed entity.
  - FBX One .FBX file for each *From...* dialog box.
  - GPH One .GPH file for each graph.
  - HLP WorkStaion help files.
  - ICN One .ICN file for each icon type used in the graphs.
  - LIS .Two .LIS files per module (the entity list, the list of the metamodels used by the module)
.Three additional files: RELATIONAL.LIS, EXTMQ.LIS and MAQLO.LIS.
  - LOG The .LOG files for the host interface.
  - MTM One .MTM file for each metamodel sub-schema.
- \SPAC\BASE: this directory includes:
  - The BASE.LST file which contains the Databases which will be proposed to the user when he/she will connect to the WorkStation,

- The connection (.SCI) and disconnection (.SCF)[1] scripts allowing access to (or exit from) Databases present on the host. NOTE: the contents of these files are specific to your host site(s) operation parameters. You must update their contents according to these parameters. For complete information, refer to Chapter *WorkStation Customization*, Subchapter *Connection and Disconnection Scripts.*
- The .PRO parameter files memorizing the user preferences. In general, these files should be neither modified nor deleted.

The BASE directory is also used as the root directory for the user data subdirectories which will be created as the user works with the WorkStation.

1.3.1. Various Installations

You can distribute the components of the WorkStation depending on the architecture of your hardware, and on the work organization you want to implement. The types of components of the WorkStation and their location in the directories are the following:

– Application executable files, in the *NNNL* directory (where *NNN* corresponds to the release number and *L* corresponds to the language);

– Methodology files, in the MET*L**MMM* directory and its subdirectories (where *MMM* corresponds to the methodology and *L* corresponds to the language);

– The user files, in the BASE directory and its subdirectories.

You can install the WorkStation on a PC only, on a network or in a mixed way (one part on PC and one part on the network). The installation of these directories can be done according to the following configurations (the following list is not exhaustive):

- Installation of all the directories on the same hard disk, a PC disk or a local network server disk. With this configuration:
  - On the PC, all the WorkStations are independent.
  - On the network, there will be only one WorkStation shared by all the users. All the PCs which have access to this WorkStation must have the same communication system with the server. Important: concurrent access problems can occur, because the user files (BASE directory and its subdirectories) are common to all users. The possible access conflicts are managed by the local network manager.

- Installation of the "non-user" part (that can be protected by the system if necessary) on the network and of the "user" part (BASE directory and its subdirectories) on the PC.
  With this type of configuration, the user data is independent and the concurrent access problems are therefore reduced. Also in this case, all PCs must have the same communication system with the server.

[1] This extension is not displayed under Windows NT

- Installation of the "non-user" part on the network, and of the "user" part plus some specific executable files on the PC. With this configuration, you can store on one disk the executable files that are common to all the WorkStations, and on another disk the executables specific to each WorkStation. The differences between WorkStations (between the communication systems for example) can be managed this way.

  We call:

  - "Executable disk" the disk that contains the executable files specific to one WorkStation.
  - "Default executable disk" the disk that contains the part common to all the WorkStations.

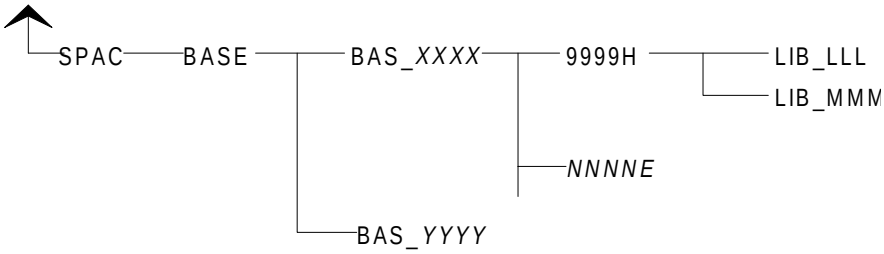
  With such a configuration, the first disk accessed is the "executable disk" (non default). If some executables are not found there, the default disk is then accessed.
  PEXEC.EXE and CONNX.EXE are used for the start-up of the WorkStation. They must stay on the "executable disk" (non default).

  According to the chosen configuration, you must adjust the parameters for the "executable disk", the "default executable disk", the system disk and the data disk in the PACBASE.DAT file. The corresponding files must be copied or moved manually, in conformity with the modified values of these parameters.

  For complete information on the PACBASE.DAT file, consult the corresponding paragraph in Chapter *Starting Up the WorkStation*.

1.3.2. Structure Evolution: User Network

When using the WorkStation, the part of the tree located under the \SPAC\BASE directory is automatically complemented with the databases, sessions and libraries accessed by the user.



The user network, using the \SPAC\BASE directory as a root, develops in the following way:

- For each new Database onto which the user connects, a subdirectory is created under \SPAC\BASE, called BAS_*XXXX*, where *XXXX* is the code of the connection Database. For example, if the user accesses the Database D280, a subdirectory BAS_D280 is automatically created if it does not already exist.

Theoretically, the list of the Databases which can be accessed and which are memorized in the \SPAC\BASE\BASE.LST file, must be in phase with the BAS_*XXXX* directory list. If one or several lines are eliminated from the BASE.LST file, it is recommended to delete the corresponding directories which serve no purpose. IMPORTANT: Before doing this operation, you must upload any user data contained in these directories to the host.

- When a user connects to a Database, he or she chooses to connect to a session and to a library.
  - Session chosen:

    If connecting to the current session, a subdirectory 9999H is added under BAS_*XXXX*.

    If connecting to a frozen session, a subdirectory *NNNNE* is added under BAS_*XXXX*.
    NNNN takes the number of the frozen session;
    E takes the value H or T (frozen type).
  - Library chosen: For a given session and for each library to the user accesses, a subdirectory LIB_*LLL* is created (where *LLL* is the accessed library code). This subdirectory is placed under the 9999H or *NNNNE* directory corresponding to the connection session. The LIB_*LLL* directories contain all descriptions of occurrences that have been downloaded from the server for local consultation or modification, as well as all drafts created under the WorkStation:
    – An *EEEDz*.LST file by local entity description type, containing the list of occurrences present on the PC in this directory. *EEE* is the local entity code and *Dz* is the local description type code (example: TXTD1.LST, where TXT is the local entity code of the Text entity, and D1 is the local code for the description type "text contents" of this entity). It is this file which allows the updating of the local lists proposed by the WorkStation (for more details about local lists, see the *WorkStation Reference Manual*).
    – A *OOOOOODz.EEE* file by for each downloaded entity occurrence. This file contains the *Dz*-type description of the *OOOOOO* occurrence of entity *EEE* (example: PTMA00D1.TXT contains the local D1-type description of the PTMA00 occurrence of the Text entity, whose local code is TXT).

The WorkStation does not automatically delete the directories that correspond to sessions and/or libraries that are not longer used. From time to time a purge of the directory tree must be done in order to avoid unnecessary directories. IMPORTANT: The same precautions must be taken as when deleting a BAS_*XXXX* directory (see above).

1.4. PC Installation – Introduction

1.4.1. Installation media

The PC part of the WorkStation is supplied on a CD-ROM. The executable files are in compressed format, but are automatically decompressed during installation.

1.4.2. Installation environment

The installation of the PC part of the WorkStation can be done on different environments, according to the desired operating environment.

Operating system (operation)	Graphical User Interface (GUI)	Installation under Windows via the Programs Manager
DOS	Windows 3.1 or 3.11 Windows 95 Windows for Workgroups	*SETUP.EXE*
Windows NT		
OS/2 2.1 WARP	Win-OS/2 (similar to Windows 3.1)	

1.5. PC Installation –DOS, WINDOWS and WIN-OS/2 Releases

1.5.1. Install

The installation can be executed directly from the installation CD-ROM or from a network directory in which the content of the CD-ROM has been previously copied.

To start installing the WorkStation or a methodology:

- insert the CD-ROM in the disk drive and wait a few seconds. The first window displayed invites you to select the installation language, and the second window invites you to select the product you want to install. Select WorkStation (Pacdesign, Pacbench, Methodology).
- or in the file manager, position to the CD-ROM drive, which contains the installation files, and select directory pdnnn (nnn represents the WorkStation release number), and then:
  - Double-click on the *SETUP.EXE* file. In the window which opens, you must specify the installation language;
  - select the eng (for English) or fra (for French) sub-directory and double-click on the SETUP.EXE file.

☞ You can stop the installation procedure at any time by clicking the "Exit" icon on the bottom right hand side of the screen or the F3 function key or the Cancel button on the bottom of each window.

1.5.2. Selecting the WorkStation and/or the Methodologies to be installed

The next window displays a list which includes the WorkStation and the various methodologies. You select one or more lines in this list, according to your needs.

If you select the WorkStation and at least one methodology, the WorkStation installation will be automatically followed by the methology installation(s). If you select more than one methodology, they will be installed one after the other, in the order of the list.

If you select the installation of "another methodology", you must have its installation diskette.

The WorkStation must be installed before the methodologies.

To select a line in the list, click on the left part of the list with the mouse's left button. To unselect it, click on the selection mark to make it disappear.

To install the WorkStation and all the methodologies listed, click the "Select all" button.

The "Clear all" button enables you to unselect all selected lines.

The enhancements of the installed release are displayed in the next window. They are stored:

- in the NEWSA.TXT (English) or NEWSF (French) file located in the UTIL directory for the WorkStation enhancements,
- in the metA.TXT or metF.TXT file ('met' is the methodology), located directly under the root of the methodology for the methodology enhancements.

Moreover, at the end of the installation, you will be able to create a "News" icon in the WorkStation/Methodology group to enable the user to view, at any time, the enhancements of the installed release.

1.5.3. Installing a new WorkStation

The first two choices of this window are important if you want to install a WorkStation on a PC where a WorkStation release is already installed.

If you select "New install", the old installation will be kept, if it is not located on the same directory as the new one.

If you install the new WorkStation in the same location as the old one, the new installation will override the old one, without any warning message.

If you select "Replace", you must specify the full access path to the executable files of the WorkStation you want to replace. You will then be able to change the location of some files if you want to.

You can select "Use save" if the WorkStation installation parameters were saved in a previous installation (see paragraph *Recapitulation of parameters*). This choice enables you to retrieve these parameters only by indicating the disk drive and the full access path to the SPACSAVE.PRM backup file. All the parameters will then be filled in automatically.

1.5.3.1. Path to the executable files, method data and user data

By default, the installation utility proposes you to install all the files on the C disk drive. But you can modify one, two or the three access paths by selecting the corresponding lines (click with the mouse's left button in the box located at the beginning of the line).

If you click "Next", selection windows enable you to specifiy the new access paths.

☞ The \SPAC WorkStation directory is automatically added to the access paths you specify; so you must not include it in the access paths.

1.5.3.2. Communication parameters

You must then specify the communication parameters, which are:

- Host type: the WorkStation host (for example MAINFRAME);
- Communication protocol under which this host operates (for example IBM 3270 or BULL DKU71xx);
- The variant of the communication protocol: you must choose from the list depending on the server. (For more information on the types of communication refer to the *Communications Manager and Paclink Utility* Manual);
- Connection mode: this parameter allows you to set the WorkStation connection modes authorized to access the WorkStation. You have the following two possibilities:
  - "PC/Host" connection,
  - "Host only" connection.

  The maximum authorization is the "PC/Host connection" which allows the user to choose between a local connection in which he/she can work with local data (already downloaded) or a host connection. The "Host only connection" requires to connect to the Host. In that case, the user recognition files are not created on the PC.

1.5.3.3. Recapitulation of parameters

Once all the parameters are entered, a summary of the selected configuration is displayed in a window, which allows you to check for errors and possibly resume the parameterization step.

You can choose to save the parameters for next installations in order to ensure the same configuration for each PC. They will be proposed again at the next installation time, so that you will not have to re-enter them.
To save the parameters, click the "Save" button. In the selection window which is displayed then, specify the disk drive and the full access path to the parameter backup file (SPACSAVE.PRM). If the installation medium is not the CD-ROM, the backup directory proposed is that which contains the SETUP.EXE program.

If you want to modify some parameters, click "Modify". You can then scroll all the installation procedure windows and correct any erroneous parameters.

If the parameters fit, click "Install" to start up installation.

If the installation is correct, a message informs you of its correct ending.

1.5.3.4. Creating icons

At the end of the installation procedure, a window displays the list of the Program Manager folders that are already installed. Select a folder in which you wish to insert the WorkStation icons ("VisualAge Pacbase WorkStation" by default) or enter a new name in the input field.

A dialog box then asks you whether you want to install the corresponding group.

If you choose "Yes", you must select the icons you want to display in this group:

- News: text file containing the new functionalities of the installed release
- PACBASE.DAT: file that contains the parameters necessary to run the WorkStation (refer to chapter *Starting Up the WorkStation*).
- WorkStation: WorkStation start-up icon
- KillWorkStation: WorkStation close icon
- PACLINK: terminal emulator start-up icon. (For details, refer to the *Communications Manager and Paclink Utility* Manual).

1.5.4. Installing methodologies

A methodology must be installed in order for the WorkStation to work.

Before installing a methodoloy, you must make sure that the WorkStation is installed already.

1.5.4.1. Selecting the methodology to be installed

The installation of the methodologies can be implemented at the same time as the WorkStation installation or independently from it. It will thus be easy to add or remove some of them.

Refer to paragraph *Selecting the WorkStation and/or the Methodologies to be installed* for more information.

1.5.4.2. WorkStation path

You must specify the full access path (including \SPAC\NNNL) to the PEXEC.EXE file.

If you specify only part of the access path, the installation ends with an error message.

If the methodology installation parameters were saved in a previous installation (see paragraph *Recapitulation of parameters*), you can retrieve them only by indicating the disk drive and the full access path to the METHSAVE.PRM backup file. All the parameters will then be filled in automatically.

1.5.4.3. Methodology path

According to the WorkStation access path specified in the previous window, the intallation utility proposes you a default access path to the methodology, which you can modify.

If the methodology does not exist in the chosen path, the next window displays "New install".

If the methodology is already installed in the chosen path, the next window displays the access path to the previous installation of the methodology. If you want to replace a methodology installed elsewhere, you must specify the full access path to the old methodology.

1.5.4.4. Recapitulation of parameters

Once all the parameters are entered, a summary of the selected configuration is displayed in a window, which allows you to check for errors and possibly resume the parameterization step.

You can choose to save the parameters for next installations in order to ensure the same configuration for each PC. They will be proposed again at the next installation time, so that you will not have to re-enter them.
To save the parameters, click the "Save" button. In the selection window which is displayed then, specify the disk drive and the full access path to the METHSAVE.PRM parameters backup file. If the installation medium is not the CD-ROM, the backup directory proposed is that which contains the SETUP.EXE program.

If you want to modify some parameters, click "Modify". You can then scroll all the installation procedure windows and correct any erroneous parameters.

If the parameters fit, click "Install" to start up installation.

If the installation is correct, a message informs you of its correct ending.

A the end of the installation, a dialog box invites you to select the modules you want to install: Pacdesign and/or Pacbench.

1.5.4.5. Creating icons

At the end of the installation procedure, a window displays the list of the Program Manager folders that are already installed. Select a folder in which you wish to insert the methodology icons or enter the name of a new folder.

A message box then asks you whether you want to install the corresponding group.

☞ if you have created a customized group for the WorkStation installation, it will be displayed in the list of the groups proposed for the methodology.

If you choose "Yes", you must select the icons you want to display in this group:

News: text file containing the new functionalities of the installed release

PACBASE.DAT: file that contains the parameters necessary to run the WorkStation (refer to chapter *Starting Up the WorkStation*).

If the installation was successful, a message informs you of its correct ending.

1.6. PC Parameterization – DOS-Windows 3.x

- The CONFIG.SYS file: Do not overload this file. For each request, a memory-resident application is loaded, thus reducing the amount of conventional memory available. However, a fine tuning of this file improves the performance of the WorkStation. Here are some of the rules to follow:
  - SHELL=C:COMMAND.COM /E:512 /P
  - FILES=20 and BUFFERS=10 are sufficient values for the operation of the WorkStation.

- SmartDrive, which is automatically installed during the Windows 3.x installation, must be configured entirely in Extended mode, which explains the absence of the EMM.SYS file.

SmartDrive is faster and... "smarter" than a virtual disk: it manages its memory space according to file access frequencies. With this system, the first access to a file (whether executable or data file) is done on the hard disk. This is relatively slow but since the file is simultaneously loaded onto the SmartDrive, all subsequent accesses will no longer require a disk access.

- The AUTOEXEC.BAT file: some adjustments for the WorkStation are done by adding the following parameters to this file:
  - SET TEMP=X:\PACTMP allows you to indicate that a PACTMP directory (to create if it does not already exist) must contain the temporary file generated by the WorkStation. X: represents a partition of the hard disk drive.
  - **Access conflicts**: To avoid all access conflicts, the SHARE option must be active. It is recommended to include this option in the AUTOEXEC.BAT file for automatic execution. You can also include it in the CONFIG.SYS file (refer to the DOS documentation).
    IMPORTANT: The WorkStation start-up does not check whether the SHARE option is active or not.
  - If the '$' character is not available on the keyboard, SET DOLLAR=*NNN* (where *NNN* is the ANSI value of the character desired). The WorkStation and the host use the '$' character for certain operations: control character in the Text descriptions, character that calls host occurrence list on U.E.O.s, etc.).

1.7. PC Parameterization – DOS-Windows 95 and NT

As for any Windows application, the installation creates a program group specific to the WorkStation. You can use the shortcuts created in this way to start up the WorkStation.

1.8. PC Parameterization – OS/2 (from 2.1 onwards) - WIN-OS/2

- The CONFIG.SYS file: add one parameter to this file:
  - SET TEMP=X:\PACTMP allows you to indicate that a PACTMP directory (to create if it does not already exist) must contain the temporary file generated by the WorkStation. X: represents a partition of the hard disk drive.

NOTE:

In OS/2 2.1, the WIN-OS/2 interface is identical to Windows 3.1.

2. WorkStation Customization

2.1. Connection and Disconnection Scripts

2.1.1. Connection modes

The WorkStation can be used in two ways:

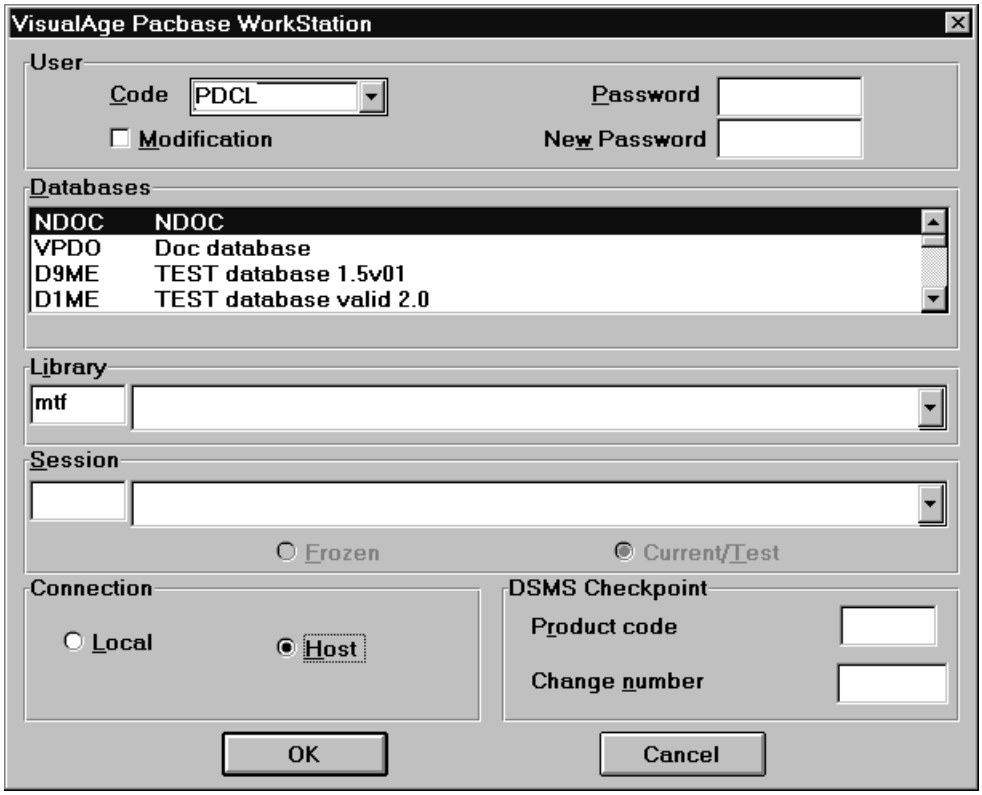
- connected to the host (host mode),
- independent of the host (local mode); it is the default mode.

☞ During the installation you have the option to allow only the host mode connection.

In general, except for the initial connection, you can connect in local mode and then switch to host mode. (See the *WorkStation Reference Manual*.) When switching to the host mode (during the WorkStation start-up or later) a host connection procedure is automatically executed.

In order for the local mode to be available, a first host connection is necessary. The user codes and passwords are then known at the local level and can be checked without a host connection.

Through the communication drivers, the WorkStation automatically connects the PC to the host for one of the cataloged Databases. The connection box is shown below:



In order for a Database name to be displayed in this connection box, it must be cataloged in the \SPAC\BASE\BASE.LST file.

To perform a host connection to a cataloged Database, you must write a connection script for this Database. A disconnection script must also be written in order to exit from this Database.

2.1.2. Cataloging a database

Each accessible Database must be cataloged as a line in the \SPAC\BASE\BASE.LST file. This line is formatted in the following way:

- Database code: this code is composed of one to four characters according to the platforms (no blank spaces, no "." or "*****") and must begin in the first column of the file.

- The Database title: starting in the fifth column of the file, you may input a label which will be displayed to the user for Database identification.

To catalog a new Database, it is necessary to add (with the help of a text editor) a corresponding line in the \SPAC\BASE\BASE.LST file.

If a line of the file is deleted, the corresponding Database will no longer be displayed in the connection box.

Example: in order to obtain the connection box previously shown, it is necessary to enter the following information in \SPAC\BASE\BASE.LST:

```
NDOC NDOC
VPDO DOC database
D9ME TEST database 1.5V01
D1ME TEST database valid 2.0
```

2.1.3. Connection scripts

The connection procedures vary from site to site depending on the systems and software in use, especially when a site has several different systems, or software managing the confidentiality and protection of information.

An interpreted language is used to write scripts which perform the connection and disconnection. This language is described in the Appendix.

For each *BBBB* Database listed in the \SPAC\BASE\BASE.LST file, two associated script files must exist in the \SPAC\BASE directory:

- A connection script, whose name is *BBBB*.SCI. This script must contain the instructions necessary to establish the connection to a TP monitor (CICS, IMS, TDS,...). It must also contain the instructions necessary to send the VisualAge Pacbase transaction code allowing the display of the sign-on screen, which will be filled in by the WorkStation.

- A disconnection script, whose name is *BBBB*.SCF[1]. This script must contain the instructions necessary to exit VisualAge Pacbase and, if necessary, the TP monitor.

[1] This extension is not displayed under Windows NT

Example: the connection script D280.SCI and the disconnection script D280.SCF correspond to the D280 Database.

Script examples are supplied in the \SPAC\BASE directory (AX_...SCI and AX_...SCF files for English scripts ; FX_…SCI and FX_…SCF for French scripts). Other examples are given in specific chapters of this manual.

NOTE: If you are connected via Top Secret, the connection and disconnection procedures are performed without using the "Script" files .SCI and .SCF.

2.1.4. Practical Suggestions

It is helpful to include, at the beginning of the connection script, an initial disconnection phase, to allow access to the WorkStation regardless of the initial state of the terminal (connected to the host or not).

It is very important to observe host response time, in order to determine the timer values. Increase the amount of testing, especially when the response time is longer than normal.

2.1.5. VisualAge Pacbase connection (OS/2, Windows NT, UNIX server)

The "actual" connection and disconnection are assured, via an OS/2, Windows NT or UNIX server, by file assignments. For complete information concerning these files, refer to the *Communications Manager and Paclink Utility* Manual.

The connection and disconnection script files must exist because the WorkStation checks their presence and tests their validity. However, as the connection/disconnection operations do not depend on these scripts, they are very short (see the examples in paragraph "Script Examples").

2.1.6. Script modes

If a Database has been cataloged properly in the PACBASE.DAT file and the associated scripts has been created, the Database can be accessed via the connection box in the following ways:

- In test mode: the majority of the instructions contained in the scripts are displayed on the screen as they are used which allows the detection of errors.

- In execution mode: the scripts, which have been tested and judged correct, are executed without being displayed on the screen.

In order to change the mode (with any text editor), edit the \SPAC*NNNL*\PACBASE.DAT file (*NNN* being the installed release number and *L* the language code) and change the 011 line, labelled "Connection execution mode", in the following way:

- 011 Connection execution mode [T] for the test mode.

- 011 Connection execution mode [E] for the execution mode.

2.1.7. Test Mode

When connecting to or disconnecting from the WorkStation, the associated script is executed and displayed in a dialog box in the following manner:

- The "connection in progress" message is displayed during the entire script execution.
- Underneath the "connection in progress" message the processing statement that corresponds to the label associated to the last executed OUTPUT command in the script is displayed.
- All the script commands are displayed in the box located under the two previous messages: the executed commands are visualized, but the structures (IF, WHILE, FOR etc.) are not.
- At the end of the script, the "script execution failed" message is displayed if an error is detected. If no error, an OK push button allows connection to or disconnection from the WorkStation.

2.1.8. Execution mode

Once the test phase is completed, the scripts can be used in execution mode, in the following way:

- The "connection in progress" message is displayed during the entire script execution.
- Underneath the "connection in progress" message is displayed the processing statement that corresponds to the label associated to the last executed OUTPUT command in the script.
- At the end of the execution of the connection script, if there was no error, the WorkStation Manager window is automatically opened. If a user or system error occurs, the script execution is interrupted and the connection box is redisplayed. If an error occurs during the checking of the host access authorization, changes can be made in the connection box and the connection will resume at the access authorization level.

2.1.9. Display Host Screen

DISPLAY commands can be added In the scripts to display the image of the host screen (25 x 80 screen), which can help when debugging the script.

The script execution is suspended, following a DISPLAY command, until the user closes the screen (via the System menu).

IMPORTANT: If you use a WRITE command (data writing to the host) in a script, followed by a DISPLAY command, the DISPLAY command erases the data written by WRITE. In order not to lose written data, the DISPLAY command must be placed *after* the SEND command which sends data to the host.

It is advisable, at the switchover from the test mode to the execution mode, to modify the scripts in order to eliminate all the test DISPLAY, except those that occur due to user or system errors (like CICS or TDS not available, disabled host transaction, etc.).

2.1.10. IBM-CICS Script examples

- Connection: Obtains the St. Marc sign-on screen, erases this screen, enters the desired CICS code. Waits for the Welcome screen, enters the user code and the password in the script dialog box, erases the Welcome screen. Sends the CICS sign-on transaction code (CESN), enters the input in the CICS transation grid, waits for the ***** screen and connects to VisualAge Pacbase. The following script is specific to a given site, and must be adapted for each site:

```
prog V280SCI
integer cr ln cl
integer cpt
string USER PASS NEWPASS

begin

OUTPUT ("Sign-on Screen", 12)
ln = 0
cl = 0
SEND ("REST")
cr = SEARCH (ln, cl, "WELCOME", 9, 2)
if (cr <> 1)
 begin
        SEND ("CLS")
        WRITEC ("CESF", 0)
        SEND ("ENT")
        ln = 0
        cl = 0
        cr = SEARCH (ln, cl, "WELCOME", 9, 3)
        if (cr <> 1)
               begin
                      cr = SEARCH (ln, cl, "UNKNOWN", 7, 2)
                      if (cr <> 1)
                             begin
                                    ERROR ("Host problem", "You
must already be connected", 1)
                                    DISPLAY ( )
                                    EXIT (12)
                             end
               end
 end

OUTPUT ("Connection to CICS", 12)
SEND ("REST")
SEND ("CLS")
WRITEC ("CICST", 0)
SEND ("ENT")
ln = 0
cl = 0
cr = SEARCH (ln, cl, "WELCOME", 7, 25)
if (cr <> 1)
 begin
        ERROR ("HOST error", "CICST not available", 1)
        DISPLAY ( )
        EXIT (12)
 end

cr = 0
cpt  = 1
while (cr <> 1)
begin
 if (cpt > 3)
        begin
               ERROR ("Number of attempts limited to 3", "Invalid
Signon ", 1)

               EXIT (15)
        end
```

```
 OUTPUT ("Signon CICS", 12)
 INPUT (System Identification","CICS User",USER,8,1, "CICS
Password", PASS, 8, 0, "New password", NEWPASS, 8, 0)
 SEND ("CLS")
 if ( USER == "ex" or USER == "EX" )
        begin
               OUTPUT("Disconnection from CICST",12)
               WRITEC("CESF", 0)
               SEND ("ENT")
               EXIT(16)
        end
 WRITEC ("CESN", 2)
 SEND ("ENT")
 ln = 0
 cl = 0
 cr  = SEARCH (ln, cl, "PASSWORD" 8,12)
 if (cr <> 1)
        begin
        ERROR ("CICS problem", "CESN not available", 0)
        EXIT (15)
        end
 WRITE (5, 25, USER, 1)
 WRITE (7, 25, PASS, 3)
 if (NEWPASS <> "                ")
        begin
        WRITE (7, 63, NEWPASS, 3)
 WRITE (9, 63, NEWPASS, 3)
        end
 SEND ("ENT")
 ln = 0
 cl = 0
 cr  = search (ln, cl, "EXPIRED", 7,3)
 if (cr == 1)
        begin
        ERROR ("Changing password", "Enter your new password", 0)
        cr = 0
        end
 else
        begin
        ln = 0
        cl = 0
        cr  = SEARCH (ln, cl, "********", 6,3)
        cpt = cpt + 1
        end
end

OUTPUT ("Connection to VisualAge Pacbase", 12)
SEND ("CLS")
WRITEC ("V280", 0)
SEND ("ENT")
ln = 24
cl = 8
cr = SEARCH (ln, cl, "CH:", 3, 12)
if (cr <>1)
 begin
        ERROR ("Problem on HOST", "VisualAge Pacbase is down”, 1)
        DISPLAY ( )
        PAUSE (3)
        EXIT (16)
 end
end
```

• Disconnection:

```
PROG V280SCF
BEGIN
 OUTPUT ("Disconnection from VisualAge Pacbase", 12)
 SEND ("CLS")
 OUTPUT ("Disconnection from CICST", 12)
 WRITEC ("CESF", 0)
 SEND ("ENT")
END
```

2.1.11. Bull-DPS7 Script examples

- Connection: Obtains $*$ by a series of transmits, waits for the input request of the TDS sign-on, sends the CN including the user code and password, waits for the acceptance map that includes the "Ready" string, sends the VisualAge Pacbase transaction code and waits for the "CH:" string of the VisualAge Pacbase sign-on screen. The following script is specific to a given site, and must be adapted for each site:

```
INTEGER CR LN CL
INTEGER CPT

BEGIN
OUTPUT ("GET PROMPT $*$", 12)
CR=0
CPT=0
WHILE (CR <> 1)
 BEGIN
 IF     (CPT > 10)
 BEGIN
               ERROR ("HOST ERROR", "$*$ NOT OBTAINED",  1)
               DISPLAY ( )
               EXIT (12)
       END
 SEND ("ENT")
 LN = 0
 CL = 0
 CR = SEARCH (LN, CL, "$*$", 3, 1)
 CPT = CPT + 1
END
OUTPUT ("SIGNON TDS", 12)
CR = 0
CPT = 1
WHILE (CR <> 1)
BEGIN
 IF (CPT > 3)
       BEGIN
               ERROR ("NUMBER OF ATTEMPTS LIMITED TO 3", "INVALID
SIGNON ", 1)
               EXIT (15)
       END
 INPUT ("IDENTIFICATION TDS", "USER TDS",USER, 8, 1, "PASSWORD &
 TDS", PASS,8,0)
 SIGNON = ""
CONCAT(SIGNON, "$*$CN STMTDS4 -USER ", USER, " -PW", PASS, &
 " -STR ?A")
 WRITEM (SIGNON,0)
 SEND ("ENT")
 OUTPUT ("TDS CONNECTION", 12)
 LN = 0
 CL=0
 CR = SEARCH (LN, CL, "CONVE", 5, 3)
 IF (CR<> 1)
       BEGIN
               LN = 0
               CL=0
               CR = SEARCH (LN, CL, "READY", 5, 3)
       END
 CPT = CPT + 1
END
OUTPUT ("CONNECTION TO VisualAge Pacbase", 12)
WRITEM ("PB00 /192", 0)
SEND ("ENT")
LN = 24
CL=8
CR = SEARCH (LN, CL, "CH:", 3, 45)
IF (CR<> 1)
 BEGIN
 ERROR ("PROBLEM ON HOST", "VisualAge Pacbase DISABLED", 1)
 DISPLAY ()
```

```
 EXIT(16)
 END
END
```

- Disconnection:

```
INTEGER RETCD LN CL

BEGIN
 OUTPUT ("DISCONNECTION FROM VisualAge Pacbase", 12)
 WRITE (24, 5, "FT", 0)
 SEND ("ENT")
 LN = 24
 CL = 8
 CR = SEARCH (LN, CL, "CH:", 3, 45)
 WRITE (24, 5, "FT", 0)
 SEND ("ENT")
 OUTPUT ("DISCONNECTION FROM TDS", 12)
 LN = 0
 CL = 0
 CR = SEARCH (LN, CL, "CONVE:", 5, 20)
 WRITEM (3$*$LO", 0)
 SEND ("ENT")
END
```

2.1.12. Bull-DPS8 Script examples

- Connection: Obtains $*$ by a series of transmits, sends the CN to the appropriate machine and TP, waits for the Logical id input request, sends Logical id, sends the VisualAge Pacbase transaction code and waits for the "CH:" string of the VisualAge Pacbase sign-on screen. The following script is specific to a given site, and must be adapted for each site:

```
INTEGER CR LN CL
INTEGER CPT

BEGIN
OUTPUT ("GET PROMPT $*$", 12)
CTP=0
CR=0
WHILE (CR==0)
 BEGIN
        IF     (CPT > 6)
               BEGIN
                      ERROR ("HOST ERROR", "$*$ NOT OBTAINED",  1)
                      DISPLAY ()
                      EXIT (12)

               END
        SEND ("ENT")
        LN = 0
        CL = 0
        CR = SEARCH (LN, CL, "$*$", 3, 3)
        CPT = CPT+1
 END
OUTPUT ("CONNECTION _ TP8", 12)************* ***
WRITEM ("CN-SC XXXX -MB PC", 0)
SEND ("ENT")
LN = 0
CL = 0
CR = SEARCH (LN, CL, "ID", 2, 15)
IF (CR <> 1)
 BEGIN
        ERROR ("HOST ERROR", "TP8 DISABLED", 1)
        DISPLAY ()
        EXIT (12)
 END
DISPLAY ()
OUTPUT ("SEND LOGICAL ID", 12)
WRITEM ("P001", 0)
SEND ("ENT")
LN = 0
CL = 0
CR = SEARCH (LN, CL, "CONNECT", 7, 12)
IF (CR<> 1)
 BEGIN
        ERROR ("HOST ERROR", "INVALID  ID", 1)
        DISPLAY ()
        EXIT (12)
 END
DISPLAY ( )
OUTPUT ("CONNECTION TO VisualAge Pacbase", 12)
WRITEC ("PB00", 0)
SEND ("ENT")
LN = 24
CL = 8
CR = SEARCH (LN, CL, "CH:", 3, 3)
IF (CR<> 1)
 BEGIN
        ERROR ("PROBLEM ON HOST", "VisualAge Pacbase DISABLED",
1)
        DISPLAY ()
        EXIT (16)
 END
END
```

- Disconnection:

```
INTEGER CR LN CL

BEGIN
 OUTPUT ("DISCONNECTION FROM VisualAge Pacbase", 12)
 WRITE (24, 5, "FT", 0)
 SEND ("ENT")
 LN = 24
 CL = 8
 CR = SEARCH (LN, CL, "CH:", 3, 45)
 WRITE (24, 5, "FT", 0)
 SEND ("ENT")
 OUTPUT ("DISCONNECTION FROM TDS", 12)
 LN = 0
 CL = 0
 CR = SEARCH (LN, CL, "CONVE:", 5, 20)
 WRITEM ("$*$LO", 0)
 SEND ("ENT")
END
```

2.1.13. OS/2, Windows NT or UNIX Script examples

Connection

```
PROG PLANSCI
BEGIN
END
```

Disconnection

```
PROG PLANSCF
BEGIN
END
```

2.2. Parameterization of Keys Equivalent to PF11 and PF12

In some cases, listed in the tables below, you must modify some lines of EMPAC.PRO file in order to activate the keys which have the same function as PF11 and PF12.

This file, automatically created at the Emulator start-up in the \SPAC\BASE directory, contains the parameterization of function keys in columns 9, 10 and 11.

You must modify the three figures of columns 9, 10 and 11 according to the configuration type of your keyboard:

- If you work on a French WorkStation and if your keyboard is not configured in the French way;
- If you work on an English WorkStation earlier than the 1.2 V00 release and if your keyboard is not configured in the French way.

NOTE "----" means that you have nothing to change.

Line number	Default value	U.S.A G.B. Canada	Germany Switzerl.	Spain Italy Portugal	Norway Sweden Finland Denmark	Netherl.	Belgium
24	219	189	----	----	187	----	----
25	187	----	221	221	219	191	189
36	219	189	----	----	187	----	----
37	187	----	221	221	219	191	189

For the English WorkStations from the 1.2 V00 release on, with a keyboard whose configuration is not English or American, you must modify the columns 9, 10 and 11 as follows, according to the configuration type of your keyboard:

Line number	Default value	France	Germany Switzerl.	Spain Italy Portugal	Norway Sweden Finland Denmark	Netherl.	Belgium
24	189	219	219	219	187	219	219
25	187	----	221	221	219	191	189
36	189	219	219	219	187	219	219
37	187	----	221	221	219	191	189

WARNING The first or the first two lines of EMPAC.PRO file are different from the other lines and are optional. One or the other or both can be absent. In such a case, take away one or two from the line numbers mentioned in the two tables above.

3. Starting Up the WorkStation

3.1. The PACBASE.DAT File

The PACBASE.DAT file contains all of the parameters which are necessary for the WorkStation start-up. This file is automatically created during the installation in the \SPAC*NNNL* directory (where *NNN* is the release number and *L* the language code of the installed release).

Each of the lines that make up this file has the following structure:

- An identifying number composed of three characters, on position 1 to 3.
- the label, whose position is free.
- the parameter value, between square brackets ([and]), whose position is also free.

Here is an example of the PACBASE.DAT file:

```
001 WorkStation Release               [250A]
002 Host                              [PACBASE]
003 Communication board               [GSEXTW]
004 Communication parameters          [GSEXTW]
005 Operating System                  [DOS]
006 Methodology                       [MER]
007 Load-module disk drive            [C]
008 Default load-module disk drive    [C]
009 System data disk drive            [C]
010 User data disk drive              [C]
011 Connection execution mode         [E]
```

The PACBASE.DAT file is created with the parameters entered during the first installation of the *NNNL* release. If the same release is re-installed without preliminary deletion, the contents of this file is not updated. The user must modify it manually (if necessary).

The type of host is coded as follows:

Mainframe	[PACBASE]
Unix	[PACBASE/X]
OS/2	[PACBASE/OS2]
Windows NT	[PACBASE/WNT]

NOTE: If you are connected via Top Secret, you must add a 012 line in the PACBASE.DAT file that indicates the access path to the local file containing these user codes, so that the WorkStation can find them.

3.2. Optional parameterization files

You can define optional parameterization files to adapt and choose the configurations of your site.

The name of these files is free but must respect the DOS file standard coding. It is advised to give them a .DAT extension.

These files must follow the pattern of the PACBASE.DAT file and be placed in the same directory as the PACBASE.DAT file.

In case of a WorkStation re-installation, the *.DAT files that have been created and the files between brackets contained in the BASE.LST file will be saved before the previous WorkStation replacement and placed back in their original directory at the end of the reinstallation.

3.3. Start-Up Procedure

You can start up the WorkStation directly from the operating system or from the Graphical User Interface (GUI) installed on the PC.

The start-up manipulations vary depending on the operation environment of the WorkStation. These differences exist only because of the differences of the operating environments. For example, to change the directory:

- under DOS you must use the CD command (Change Directory) with the directory name (ex.: CD \SPAC*NNNL*).
- under Windows, it is the File Manager which allows you to change directory, by selecting the desired directory in the directory tree displayed in a window.

In this chapter, we indicate, in general terms and regardless of the WorkStation operation environments, what needs to be done on the start-up directory and program. For the details concerning the manipulations related to your environment, please refer to the user documentation for your operating system and GUI.

3.3.1. Start-up directory: \SPAC*NNNL*

Before starting-up the WorkStation, you must be positioned on the \SPAC*NNNL* directory of disk X: on which the WorkStation has been installed. The WorkStation initialization programs must access particular files (for example, PACBASE.DAT) which are located in this directory.

The content of this directory is detailed in Subchapter *WorkStation Directory Structure: Initial state and evolution* of Chapter *WorkStation Installation*.

3.3.2. Start-up: PEXEC.EXE

To start up the WorkStation, execute the PEXEC.EXE program. This program must be run under the GUI corresponding to your installed WorkStation release: Windows, Presentation Manager or Win-OS/2.

By default, the PEXEC.EXE program takes the PACBASE.DAT parameter file into account.

However, you can specify an optional parameterization file (that must be defined):

- Give the name of this file as a parameter in the PEXEC.EXE program, in the command line:

  EXAMPLE: C:\SPAC\250A\PEXEC.EXE\PACBASE.BIS.

  The parameters of the PACBASE.DAT file then are not taken into account.

- Or, more specifically, associate the name of this file to a database in BASE.LST file. You must put the name of the file into brackets (ex: V280 Validation Database [PACBASE.TER]). This name will appear in the database list, next to the database label. These new parameters will overwrite the file parameters (PACBASE.DAT or PACBASE.BIS) which were used to start the WorkStation.

NOTE: The optional parameterization files (here, PACBASE.BIS or PACBASE.TER) must exist in the same directory as PACBASE.DAT. They must be constituted like the PACBASE.DAT (with different values in the parameters). The name of these files is free. PACBASE.DAT must not be deleted.

4. WorkStation Utilities

4.1. External Trigger

This utility enables you to start WorkStation applications without using the WorkStation menus.

Its use is recommended when you work under the control of a project management tool, such as LCM or Microfocus WorkBench.

You can start it up from the Windows Program Manager.

4.1.1. Functionalities

The Trigger only works if the WorkStation Manager is active.

It enables you to:

- access the occurrence descriptions and definitions (but not the lists and cross-references).
- open the following windows: Library/Session, Generation-Print, Emulator, Data Extraction, List of Journalized Transactions, Keyword Search, and Generation-Print Monitor (only for the OS/2, Windows NT or UNIX VisualAge Pacbase WorkStation).

4.1.2. Activation

To activate this utility, select the **Run** function in the **File** menu of the Windows Program Manager, then enter the following command line:

- To access a description:

  C:\SPAC\250A\WSTIM MODULE EEE XXXXXX D*d*,

  where:

  MODULE is the work module;

  EEE the entity code;

  XXXXXX the occurrence code;

  D*d* the description code;

  (EEE and Dd are the local codes; for the list of local codes, refer to the Pacdesign or Pacbench manual of your methodology).

- To access a definition:

  C:\SPAC\250A\WSTIM MODULE EEE XXXXXX DEF,

  where:

  MODULE is the work module;

  EEE the entity code;

  XXXXXX the occurrence code;

  DEF the definition code;

  (EEE is a local codes; for the list of local codes, refer to the Pacdesign or Pacbench manual of your methodology).

- To open a window:

  C:\SPAC\250A\WSTIM XXXXX,

  where X is the code of the application to be started (BIBSS for Library/Session, EDGEN for Generation-Print, EMPAC for Emulator, EXTMQ for Data Extraction, JOURN for List of Journalized Transactions, WSRCH for Keyword Search and GPMON for Generation-Print Monitor).

  In order to represent the various Triggers (ex: a Trigger of the Generation-Print application, another one of the List of Journalized Transactions application, ...) in the WorkStation Manager, you can select any icon of the WorkStation.

☞ If the external Trigger is activated from LCM, this latter determines the working time spent on the description and deduces the workload already performed and the remaining workload. This information is required for the project follow-up.

4.2. ILRTF Local Editing Application

The "ILRTF.EXE" local editing application replaces the "IMPLO.EXE" local printout utility and enlarges its action field. It is automatically installed with the WorkStation applications, and is stored in the SPAC*NNNL* subdirectory corresponding to the WorkStation subdirectory (English or French version).

ILRTF.PRO is the parameters file, located in the SPAC\BASE sub-directory.

If the word processor you are going to use is not an English one, you must specify, in **line 6** of the ILRTF.PRO file, the equivalent of the 'Titre' style in your word processor. This information is vital so that the word processor software recognizes this style and can number the titles in the document. If this line is missing the style name for titles is "Titre".

Otherwise and for your information, the ILRTF.PRO file contains information used by the ILRTF.EXE application. When using the application for the first time, the fields of the "Converting a PDM file to an RTF file" window are empty. Then on each use the files' paths entered in each field are memorized in the ILRTF.PRO file. The memorized files will be displayed in a drop-down list for each field in the application. If the drop-down list gets too long, you can delete file paths in ILTRF.PRO.

The following lines of the ILRTF.PRO file are loaded in the drop-down lists of the following fields:

- **Lines 1** contain the paths to the files entered in the "PDM source" field.

- **Lines 2** contain the paths to the files entered in the "Style sheet" field.
- **Lines 3** contain the paths to the files entered in the "Target" field.
- **Line 4** contains the path to the PACBASE.DAT file entered in the "Workstation Configuration File" field.
- **Line 5** contains the path to the word processor file used which was entered in the "Word processor used" field in the window which shows the results of the generation.

NB: **Line 7** indicates the titles' generation mode that was specified by using the window's two radio buttons:

1 indicates a standard generation.

2 indictaes a PDM generation.

4.3. The Clean-Up Utility

A utility enables you to close all applications that could remain open after a sudden interruption of the WorkStation. When using a PC, external problems may always cause an interruption.

In this case, use the KILLP.EXE utility located in the \SPAC*NNNL*\UTIL directory.

5. Appendix: The Script Language

5.1. Introduction

The objective of the script files is to write the logon (logoff) procedures that would otherwise be entered manually each time the end-user wishes to sign-on to or sign-off from VisualAge Pacbase. This is done by a series of instructions, designed by you, and sent transparently by the WorkStation.

Connection procedures to host sites must be extremely precise. Therefore, we have provided an interpreted language where the control structures are more plentiful than those available through batch. The language and its interpreter are described in this Appendix.

5.2. Script Structure

The structure of a connection script is:

```
PROG                Script_name
declaration(s)
BEGIN
instruction(s)
END
```

The first line of a script must have the following structure:

```
PROG                Script_name
```

This line declares the script's program name.

The PROG keyword must be written in small or capital letters. The name of the program is a series of letters and of numbers beginning by a letter and containing a maximum of 16 characters. This word must NOT be a reserved word. (The list of reserved words is given below.)

No other information may be included on this line.

5.3. Reserved Words

Keyword and function names are reserved. They may not be used as names or as labels for programs and variables.

AND	BEGIN	BOOLEAN
BREAK	DO	ELSE
END	FALSE	FOR
GOTO	IF	INTEGER
NOT	OR	PROG
STRING	TRUE	UNTIL
WHILE		

The following keywords, entered in uppercase, correspond to pre-defined functions:

CONCAT	DISPLAY	ERROR
EXIT	CURPOS	CPYSCR
INPUT	INTSTR	OUTPUT
PAUSE	READ	READC
RPARAM	SEARCH	SEND
WRITE	WRITEC	WRITEM

5.4. Declarations

The declaration part of the program allows the definition of variables and labels used in the instructions. This part is located between the heading and the body of the program (if no variables are used, this part can be deleted).

- Variable declaration:

```
Variable_type    Variable_name
```

- Label declaration:

```
$label
```

5.4.1. Variable Types

There are three types of variables: integer, string and boolean.

A declaration of a variable is written as follows:

```
          STRING     Variable_name
or        INTEGER    Variable_name
or        BOOLEAN    Variable_name
```

Several variables of the same type can be declared on the same line:

```
STRING  name1 name2
```

is the same as:

```
STRING  name1
STRING  name2
```

5.4.2. Variable Names

A variable name is a series of alphanumeric characters, beginning by a letter. The maximum length is 16 characters. All variables used in the program must be declared. Two variables cannot have the same name. A variable name must not be a keyword or a function name (see the previous list for the reserved words.)

The variable names must be followed by a blank character or by another non alphanumeric character. A declaration ends with a return.

5.4.3. Variable Values

The value of a variable is indicated in the following way:

```
Variable_name=Value
```

Integers are between (-maxint) and (maxint -1) where maxint depends on the machine used (2^{15} = 32768 on IBM PC). Integers are manipulated by the operators decribed under "Instructions."

Character strings, which can include blanks, have a maximum length of 132 characters. They are entered between double quotes. When one of the characters in the string is a double quote itself, it must be doubled.

When a string does not fit on one line, it is necessary to insert a "&" before the return at the end of the line. There can only be blanks or tabulation marks between the continuation character and the end of the line, if not the "&" is considered as a character in the string. The rest of the string begins in column 1 of the next line.

Examples:

```
STRING="He says""Hello""."
The contents of the string are:
He says "Hello".

STRING="This string continues&
on the following line."
The contents of the string are:
This string continues on the following line.

STRING="This string contains a & (ampersand)."
The contents of the string are:
This string contains a & (ampersand).
```

Booleans only contain two values: TRUE and FALSE. They can be combined with the logical operators (see below). Booleans are recognized in all conditional structures.

5.4.4. Labels

The labels used for branching (GOTO $XXX) must also be declared. A label contains a maximum of 3 numbers.

Label declaration:

$label_name

No other information may be included on this line. All program labels must have distinct names.

5.4.5. Comments

Comments, placed between colons ":", are not interpreted. They may be inserted anywhere, except in character strings. It is possible to write several lines of comments by placing a ":" at the beginning of the first line and a ":" at the end of the last line. Instructions inserted between comment delimiters are not taken into account:

Examples:

```
PROG       PROGRAM_NAME
:A COMMENT MAY TAKE UP
SEVERAL LINES :
BEGIN END
```

```
PROG PROGRAM_NAME              :A PROGRAM HAS SEVERAL LINES:
BEGIN                          :OF COMMENTS IN THE MARGIN:
END
```

```
PROG PROGRAM_NAME              :THESE COMMENTS COVER THE KEY
BEGIN                          WORD BEGIN.  THIS IS AN ERROR:
END
```

```
PROG                           PROGRAM_NAME
STRING                         STRING_NAME
BEGIN                          STRING_NAME="ABCDEF:COMMENTS:"
END
```

```
The value of the string is:    ABCDEF:COMMENT:
and not ABCDEF
```

5.4.6. Blanks and Returns

Blanks are equivalent to tabulations and mark the end of a string of characters. In the remaining part of this chapter, blanks and tabulations will be underlined when they are required. A keyword that is immediately followed by an alphanumeric character is not recognized. For example, a heading cannot be written PROGname: A blank is required after keywords.

A return represents the end of an instruction. If several lines are required for an instruction, the character "&" may be used as a continuation character before the return. The "&" sign must be followed by a blank or a tabulation to be used as a continuation character, otherwise it is considered as part of the string.

Example:

```
INPUT("sign-on CICS","user code",&UTI,6,1,)
```

is equivalent to:

```
INPUT("sign-on CICS","User Code",UTI,6,1,)
```

When a return is used, it is possible to leave one or several blank lines, except in the IF ELSE control structure (see IF ELSE) in which a blank line is not allowed before ELSE.

5.5. The Body of the Program

This section is delimited by the keywords BEGIN and END. Between these two keywords is a sequence of instructions whose syntax is described in this chapter.

5.6. Instructions

The instructions, that make up the body of a program, consist of function calls, of GOTOS to other parts of the program, and of value assignments to expressions.

5.7. Assignments

Assignments involve giving a variable the value of an expression, as follows:

```
variable_name = expression
```

No other information may be entered on the line.

Expressions may contain variables, integers, literals, boolean constants, or a combination of other expressions.

5.8. Expressions and Operators

The simplest form of instructions are as follows:

- variable_name
- integer
- literal
- boolean constant

The script language provides arithmetic and logical operators that enable you to define expressions:

expression: expressionA Operator expressionB

Arithmetic operators combined with interger operators produce integers:

-expressionA	opposite of expressionA
expressionA + expressionB	sum of the expressions
expressionA - expressionB	difference of the expressions
expressionA * expressionB	product of the expressions
expressionA/expressionB	quotient of the expressions

The local operators produce boolean results (having the value TRUE or FALSE):

- exp_A < exp_B is TRUE if exp_A is less than exp_B; exp_A and exp_B must be integers.

- expA > expB is TRUE if expA is greater than expB; expA and expB must be intergers.
- expA == expB is TRUE if expA is equal to expB; expA and expB must be of the same type.
- expA <> expB is TRUE if expA is different than expB; expA and expB must be of the same type.
- expA and expB is TRUE if expA and expB are true; expA and expB must be boolean.
- expA or expB is TRUE if expA or expB is true; expA and expB must be booleans.
- not expA is TRUE if expA is FALSE and vice versa; expA must be boolean.

5.8.1. Priority of Operators

The evaluation of expressions that use several operators is done according to a predefined hierarchical order. Two rules govern this order: the hierarchy of the mathematical operator and its location in the expression.

The following operators are shown their order of priority (highest to lowest):

-	unary
* /	associative
- +	associative
< > < > ==	non associative
NOT	non associative
AND OR	associative

5.8.2. Processing of associative operators

To evaluate	A + B + C
use	intermediate = A + B
and the result is	intermediate + C

Two non-associative operators cannot be adjacent. The product of A and the sum of B and C must be written:

```
A*B+A*C
```

The parentheses "(" and ")" change the order of the evaluation. An expression between parantheses is equal to a temporary variable, the inner-most parentheses are evaluated first:

```
A=(B+C-D)*(E/F)
```

is equivalent to:

```
intermediate_1=B+C
intermediate_2=intermediate_1-D
intermediate_3=E/F
A=intermediate_2*intermediate-3
```

The parentheses also make the expression more readable without modifying the evaluation order. Their use is recommanded for expressions with operands of different types, for example:

```
(a>0) and (a<11)
```

5.9. The Unconditional Branch

The instruction GOTO $XXX
enables the program not to be executed in order, but from the location marked with $XXX.

The blank between "GOTO" and "$" is optional and a return is required at the end of the line.

All branches are not allowed: a branch cannot be used to enter into an instruction block, but one can be used to exit it.

5.10. Control Structures

Control structures consist of groups of commands whose execution is conditioned.

The control structures are:

IF condition block ELSE block	WHILE condition block UNTIL condition	DO block	FOR condition block

A condition is written:

(expression)

where the expression has a boolean value. The condition is true if and ONLY if this boolean value is TRUE.

The condition of the structure FOR is special, see below.

The keyword defining the structure and the condition must be entered on the same line, separated by a blank.

A block of instructions is used to group several instructions to be executed sequentially, together, under the same condition.

The instruction block is preceded by the keyword BEGIN and followed by the keyword END.

Branching can be done only from within a block, however, branching from outside a block to within a block is not permitted.

5.11. Expressions

A block of instructions is written as follows:

BEGIN instructions END

A return is required at the end of each line. It can be inserted before the block, after BEGIN, before and after the instructions.

5.11.1. IF ELSE

```
IF      condition
        BEGIN
        instructions
        END
ELSE
        BEGIN
        instructions
        END
```

When the interpreter detects this structure, it evaluates the condition and then executes the first block if the condition is true, the second if it is false.

The "ELSE BEGIN instruction END" is optional. In this case, the program immediately executes the sequence if the condition is false.

Only one return can be placed between the first END and ELSE.

5.11.2. WHILE

```
WHILE   condition
        BEGIN
        instructions
        END
```

The interpreter evaluates the condition, and if true, executes the block, and repeats the operation. The block is therefore re-executed, as long as the condition continues to be true. If the condition is false the first time it is tested, the block will never be executed, and the program continues sequential execution at the end of the block. Be careful to make sure that the condition eventually becomes false, or else you will create an infinite loop.

5.11.3. DO UNTIL

```
DO
        BEGIN
        instructions
        END
UNTIL   condition
```

This instruction block is executed as long as the condition is false. The first part (DO) is executed at least once. Be careful to make sure that the condition eventually becomes true, or else you will create an infinite loop.

5.11.4. FOR

```
FOR (expressionA, variable_name, expressionB)
BEGIN
Instructions
END
```

The "variable_name" acts as a counter, and must therefore be an integer. ExpressionA (must be an integer) is evaluated once, and its value defines the lower limit. ExpressionB (must be an integer) is the upper limit. The interpreter assigns the value of the lower limit to the counter and compares it to the upper limit. If the upper limit is greater than or equal to the counter, the block is executed, the counter is incremented by one, and a new comparison is made with the upper limit. If the lower limit is greater than the upper limit, the block is not executed, and the program continues sequentially, after the block.

This structure allows you to execute a block a set of number times (upper-limit - lower-limit +1). However, the counter cannot be used manually. Specifically, it must not be modified within a block, it cannot be used as a counter for a nested FOR loop, and if it is used somewhere else in the program, the old value will be lost at the beginning of the FOR loop.

5.11.5. **Exit From a Loop (break)**

As soon as the interpreter detects the keyword BREAK, it exits from the nearest FOR, WHILE or DO UNTIL loop. This allows you to get out of an iterative structure prematurely - used usually in testing. A BREAK outside a FOR, WHILE or DO UNTIL loop will provoke an error.

The keyword BREAK must be followed by a return.

5.12. Predefined Functions

Predefined functions can be called either as instructions, or as expressions. That is, after the call, some will produce a value, some will not. Those that do not produce values are designed to interact with the environment. They provide the technique to have the program communicate with the outside.

5.12.1. Predefined Function Calls

A function call is entered using the following syntax:

```
function_name (parameter_list)
```

Each function will be described later, as to whether or not a value is produced, and what the values are. The functions that produce values may be used either in an expression, or as an instruction, but the value of the return code will be lost. Functions that do not produce values can only be used instructions.

5.12.2. Parameters for Functions

There are two different types of parameters: those that may be modified by the function, and those that may not. Those that the function does not modify are evaluated before the execution of the function. Their values can be represented by expressions or variables, in the latter case, the called function will only recognize the variable's value at the actual call, and it is not modified. For those parameters that are modified by the function, variables of the expected type must be used. The function will then transform the contents of this variable, according to the specifications of the function. The details are described in the function descriptions.

A function may have 0, 1 or many parameters. Each parameter must be separated by a comma. Parameters may be INTEGER, STRING or BOOLEAN. The number of parameters and their types are tied to the specifications of each individual function and must be respected.

NOTE: In the function descriptions, the word CNT indicates a constant parameter, and the word VAR indicates a modifiable parameter. CR indicates a return.

5.13. Communications Functions

5.13.1. PAUSE

Sets a time period between two commands.

cr PAUSE (Par1)

Par1	cnt integer	time in seconds
cr	integer	1

5.13.2. SEND

Sends the value of a key.

cr SEND (Par1)

Par1		cnt string	key to send
cr	integer	1	normal send
		2	unknown F key
		3	no answer from host

KEYS FOR ALL HARDWARE:

ENT	enter
GCHE	cursor to the left
DRTE	cursor to the right
HAUT	cursor upwards
BAS	cursor downwards
TAV	tabulation (on formatted screen)
TAR	tabulation backwards (on formatted screen)
PF01 to PF24	PF keys

WITH A 3270 PROTOCOL:

ATTN	attn
ATTR	attributes
CLS	clear screen
CURS	select cursor
DEL	delete one character
DUP	duplicate
EEOF	erase EOF
ERA	erase
HOME	return to beginning of line
PA1	PA1
PA2	PA2
PRN	print
REST	restore
RSET	reset
SYST	system call

5.13.3. WRITE

Writes a character string on the screen.

cr WRITE (Parm1, Parm2, Parm3, Parm4)

Param1: cnt integer line

Param2: cnt interger column

Param3: cnt string character string to write

Param4: cnt integer MDT bit modification (3270 mode)
display the string in test mode

0 no modification of the MDT, string displayed

1 modification of the MDT, string displayed

2 no modification of the MDT, string not displayed

3 modification of the MDT, string not displayed

cr	integer	1	write ok
		2	error on write

5.13.4. WRITEC

Writes a character string on the screen at the position of the cursor.

cr WRITEEC (Parm1, Parm2)

Parm1:	cnt string	string to write
Parm2:	cnt integer	MDT bit modification (3270 mode) display the string in test mode

The second parameter, i.e. the type, allows you to modify the value of the MDT bit (3270 mode), and to display the string or not in test mode:

0	no modification of the MDT, string displayed
1	modification of the MDT, string displayed
2	no modification of the MDT, string not displayed
3	modification of the MDT, string not displayed

cr	integer	1	write ok
		2	error on write

Note that the write may not be executed when the cursor is not on an input field.

5.13.5. WRITEM

Writes a character string in non-formatted mode - for Bull hardware.

cr WRITEM (Parm1, Parm2)

Parm1:	cnt string	character string to write	
Parm2:	cnt integer	in test mode:	0=string diplayed 2=string hidden

cr	integer	1	write ok
		2	error on write

5.13.6. READ

Reads a character string on the screen.

cr READ (Parm1, Parm2, Parm3, Parm4)

Parm1	cnt integer	line
Parm2	cnt integer	column
Parm3	var string	string receiving the result
Parm4	cnt integer	length of the field to read

cr	integer	1	read ok

5.13.7. READC

Reads a field on the screen at the position of the cursor.

cr READC (Parm1, Parm2)

Parm1	var string	string receiving the result
Parm2	cnt integer	length of the field to read

cr	integer	1	read ok

5.13.8. SEARCH

Searches a character string on the screen.

cr SEARCH (Parm1, Parm2, Parm3, Parm4, Parm5)

Parm1	var integer	line where the search begins (0: the search is performed on all lines)
Parm2	var integer	column where the search begins (0: the search is performed on all columns)
Parm3	cnt string	string to search
Parm4	cnt integer	length of the string to search
Parm5	cnt integer	maximum duration of the search in seconds

cr	integer	0	string not found
		1	string found

5.13.9. CURPOS

Reads the position of the cursor on the host screen.

cr CURPOS (Parm1, Parm2)

Parm1	var integer	line number
Parm2	var integer	column number

5.13.10. DISPLAY

Displays the host screen.

cr DISPLAY ()

This causes an interruption in the script execution. It is useful in testing scripts, and in error conditions, it will allow the user to see the problem screen. The screen displayed cannot be modified.

Parameters: None

cr	integer	1

5.13.11. CPYSCR

Copies the host screen into a PC file.

This command does not interrupt the script execution; it is used mainly in test mode or in case of error, in order to allow the user to determine the cause of the error. Note that all screens copied are stored in the file, which must be deleted manually.

cr CPYSCR (Parm1)

Parm1	var string	complete name of the DOS file (U:\PATH\PREFIX.EXT).
cr	integer	1

5.14. Input-Output Functions

5.14.1. INPUT

Creates a dialog box.

```
cr    INPUT    (Parm1,ParmN,ParmN+1,ParmN+2,1ParmN+3,&
                                ParmN,ParmN+1,ParmN+2,ParmN+3,&
                                ...
                                ParmN,ParmN+1,ParmN+2,ParmN+3)
```

This function has a variable number of parameters:

Parm1	cnt string	name of the dialogue box

The number of parameters of this function depends on the number of lines defined in the dialogue box. They are entered in series of four where:

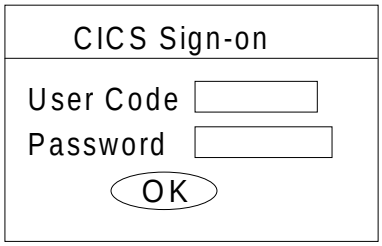
ParmN	cnt string	label of the input field
ParmN+1	var string	variable receiving the user's answer.
ParmN+2	cnt integer	maximum length of the answer (additional characters are not taken into account)
ParmN+3	cnt integer	display user input on the PC screen: 1 = yes 2 = no (e.g.: passwords)

cr	integer	1	ok

Example:

```
INPUT ("CICS Sign-on ","User Code",UTI,6,1,"Password", PSS,8,0)
```

This command builds the following dialog box:



This box includes two input fields: the user code (UTI), six characters which are displayed, and the user password (PAS), eight characters which are not displayed. The string for “CICS Sign-on” is truncated after 50 characters.

5.14.2. OUTPUT

Displays a message on the PC screen.

cr	OUTPUT (Parm1,Parm2)	
Parm1	cnt string	message to display
Parm2	cnt integer	display type
0 or > 6		the message is displayed in the connection box, in order to identify the current connection phase.
1 to 6		message box overrides the connection box; see the documentation on the ERROR function which follows for a complete description of these values.
cr	integer	1 if parameter 2 = 0. See the documentation on the ERROR function which follows for a complete description of these values.

5.14.3. ERROR

Displays a message box.

cr	ERROR (cnt string,cnt string,cnt integer)	
Parm1	cnt string	box caption
Parm2	cnt string	error message
Parm3	cnt integer	keys:

	WINDOWS - English
1	OK
2	OK CANCEL
3	ABORT RETRY IGNORE
4	YES NO CANCEL
5	YES NO
6	RETRY CANCEL

cr integer value of the key pressed by the user:

1	OK
2	CANCEL
3	ABORT
4	RETRY
5	IGNORE
6	YES
7	NO

5.14.4. RPARAM

Reads VisualAge Pacbase identification connection parameters during the script execution.

cr RPARAM (cnt integer,var string)

Parm1	cnt integer	type of parameter to retrieve
Parm2	var string	string receiving the parameter

cr	integer	1	OK
		2	invalid value for Parm1

Parameter type

0	user code
1	user password
2	database code
3	library code
4	session number (9999: current session)
5	session type (H: frozen session/T: test session)
6	DSMS Product code
7	DSMS Change number

5.15. Technological Functions

5.15.1. EXIT

Ends the script execution.

EXIT (Parm1)

Parm1	cnt integer	value different from 0 and from 1

5.15.2. CONCAT

Concatenates several character strings.

cr CONCAT (Parm1,ParmM, ... , ParmX)

Parm1	var string	variable receiving the result of the concatenation
ParmM to ParmX	cnt string	string to concatenate
cr	integer	length of the result

NOTE: if the variable used as parameter 1 is not empty, its contents are also concatenated; the length of the resulting string must be less than or equal to 132 characters (maximum length of a character string).

5.15.3. INTSTR

Converts an integer into a string.

cr INTSTR (Parm1,Parm2)

Parm1	var string	resulting string
Parm2	cnt integer	integer to convert
cr	integer	length of the result

5.16. Error Management

When the interpreter detects an error, it terminates the script's execution and sends a message which indicates the number of the erroneous line and if possible a diagnostic message.

Errors may be divided into three groups: source code errors, syntax errors and execution errors. The first two groups involve errors found in reading the source code: variable types and their utilization, branches and their labels, and syntax errors. The second category involves errors found in the source of the execution of the script.

5.16.1. Source Code Errors

The principal characteristic of source code errors is that the interpreter will send an error message to the screen should an error be detected and then continue. The script however is not executed.

The interpreter checks for compatibility between variable and constant types, the validity of expressions and of parameters. Constraints imposed upon operator types, parameters of predefined functions, counters of FOR loops and conditions are documented earlier in this chapter.

Several tests are performed on variables: using a variable which was not initialized directly or by assignment; using a variable that is used as a parameter in a predefined function causes an error, as well as using an undeclared variable. When variables are declared and not used, a warning message is issued though this causes no error.

Unconditional branching is strictly controlled: labels must be defined; branching inside an instruction block causes an error. When declared labels are not used, a warning message is issued, though this causes no error.

5.16.2. Syntax Errors

When a syntax error is encountered, the interpreter issues an error message and returns control to the user. The error message indicates the number of the line where the error was detected. Note that the line number refers to the line on which the inconsistency problem was detected, and not necessarily where the correction needs to be made.

EXAMPLE:

```
1       PROG PROGRAM_NAME
2       STRING SELECTOR
3       BEGIN
4               SELECTOR="OPEN"
5               IF (SELECTOR=="OPEN")
6                       BEGIN
7                               WRITEM ("RUN")
8                       END
9               ELSE
10              BEGIN
11                      WRITEM ("WAIT")
12                      PAUSE(25)
13      END
```

In this example, the error will be detected on line 13, whereas the error is due to a missing END at line 12: the interpreter cannot know the number of lines contained in the ELSE block before it reads line 13.

Syntax errors are often due to:

- missing or superfluous block delimiters (BEGIN END),
- operators which should be on a new line but are not,
- line skip before an ELSE,
- unmatched parentheses,
- missing parenthesis in conditions,
- final END missing.

5.16.3. Errors During Execution

Though many errors may be detected before an actual execution, there are problems which may only appear when the script is run.

These problems depend on the length and intricacies of the script, as well as on the correctness of algorithms. Problems due to limitations of the PC's memory and the maximum time allowed for routine execution depend on the hardware and operating system used: for instance, an endless loop could cause an error or may even impact on the PC's performance.

Exceeding the maximum value of an integer (32767) may also cause errors. The interpreter terminates execution when such a problem occurs. The error message may appear inaccurate since it does not always give the number of the line on which the error occurred.