VisualAge Pacbase

PAW GUIDE DU DEVELOPPEUR

DSPWO000161F

Remarque

Avant d'utiliser le présent document et le produit associé, prenez connaissance des informations générales figurant à la section "Remarques" de la page suivante.

En application de votre contrat de licence, vous pouvez consulter ou télécharger la documentation de VisualAge Pacbase, régulièrement mise à jour, à partir du site Web du Support Technique :
http://www.software.ibm.com/ad/vapacbase/support.htm

La section Catalogue dans la page d'accueil de la Documentation vous permet d'identifier la dernière édition disponible du présent document.

1ère Edition (juillet 1997)

La présente édition s'applique à :

- VisualAge Pacbase Version 2.0
- VisualAge Pacbase Version 2.5

Vous pouvez nous adresser tout commentaire sur ce document (en indiquant sa référence) via le site Web de notre Support Technique à l'adresse suivante :
http://www.software.ibm.com/ad/vapacbase/support.htm
ou en nous adressant un courrier à :
IBM Paris Laboratory
Support VisualAge Pacbase
30, rue du Château des Rentiers
75640 PARIS Cedex 13
FRANCE

IBM pourra disposer comme elle l'entendra des informations contenues dans vos commentaires, sans aucune obligation de sa part.

REMARQUES

Ce document peut contenir des informations ou des références concernant certains produits, logiciels ou services IBM. Cela ne signifie pas qu'IBM ait l'intention de les annoncer dans tous les pays où la compagnie est présente.

Toute référence à un produit, logiciel ou service IBM n'implique pas que seul ce produit, logiciel ou service puisse être utilisé. Tout autre élément fonctionnellement équivalent peut être utilisé, s'il n'enfreint aucun droit d'IBM.

Il est de la responsabilité de l'utilisateur d'évaluer et de vérifier lui-même les installations et applications réalisées avec des produits, logiciels ou services non expressément référencés par IBM.

IBM peut détenir des brevets ou des demandes de brevet couvrant les produits mentionnés dans le présent document. La remise de ce document ne vous donne aucun droit de licence sur ces brevets ou demandes de brevet. Si vous désirez recevoir des informations concernant l'acquisition de licences, veuillez en faire la demande par écrit à l'adresse suivante :

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

Les détenteurs de licences du présent produit souhaitant obtenir des informations sur celui-ci à des fins : (i) d'échange d'informations entre des programmes développés indépendamment et d'autres programmes (y compris celui-ci) et (ii) d'utilisation mutuelle des informations ainsi échangées doivent s'adresser à :

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

De telles informations peuvent être mises à la disposition du Client et seront soumises aux termes et conditions appropriés, y compris dans certains cas au paiement d'une redevance.

IBM peut modifier ce document, le produit qu'il décrit ou les deux.

MARQUES

IBM est une marque d'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 et VisualAge sont des marques d'International Business Machines Corporation, Inc. dans certains pays.

Java et toutes les marques et logos incluant Java sont des marques de Sun Microsystems, Inc. dans certains pays.
Microsoft, Windows, Windows NT et le logo Windows sont des marques de Microsoft Corporation dans certains pays.
UNIX est une marque enregistrée aux Etats-Unis et/ou dans d'autres pays et utilisée avec l'autorisation exclusive de la société X/Open Company Limited.

D'autres sociétés peuvent être propriétaires des autres marques, noms de produits ou logos qui pourraient apparaître dans ce document.

Table des Matières

PAW

PAW (**P**acbench **A**utomatic **W**indowing) est un logiciel permettant d'utiliser sur un micro-ordinateur des applications prévues pour un terminal passif, avec un plus grand confort d'utilisation.

Ces applications sont présentées via une interface graphique de type MS-WINDOWS et offrent donc les commodités habituelles de ce type d'interface (utilisation de la souris, menus déroulants, documentation interactive).

Dans la suite de ce document, nous serons amenés à utiliser les termes d'utilisateur et de développeur, dont les définitions suivent :

- l'utilisateur est la personne qui se sert habituellement de l'application. Il est équipé d'un micro-ordinateur qu'il utilise en remplacement d'un terminal passif et sur lequel le développeur a installé le paramétrage d'habillage. Nous désignerons par PAW le module dont se sert l'utilisateur.

- le développeur est la personne chargée de mettre en place le produit, de réaliser l'habillage des applications, c'est à dire de générer le paramétrage d'habillage, et de l'installer sur les postes des utilisateurs. Nous désignerons par PAWGEN et PAWLIS les modules dont se sert le développeur pour générer le paramétrage d'habillage.

Remarques à propos du manuel

Le manuel que vous avez entre les mains s'adresse à deux types d’utilisateurs : ceux qui désirent habiller les applications qu'ils ont eux-mêmes réalisées sur serveur et ceux qui désirent habiller des logiciels IBM prévus pour un terminal passif (DSMS en l'occurrence).

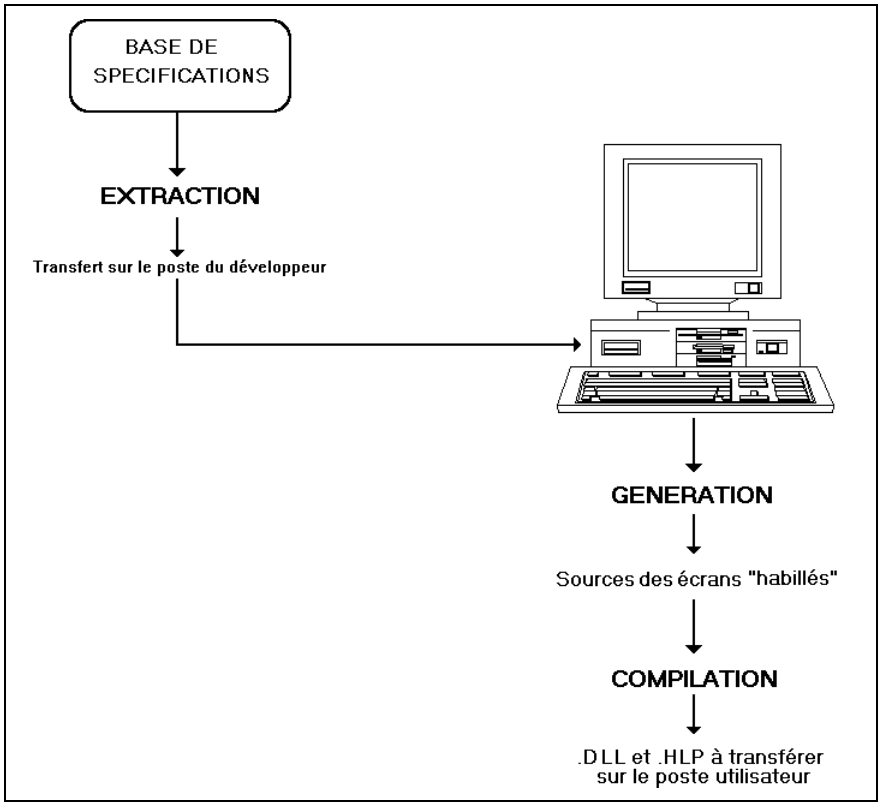
Les premiers doivent générer de toutes pièces les fichiers permettant d'habiller leurs applications avant de les installer sur l'ensemble des postes utilisateurs. Les cinq premiers chapitres s'adressent à eux.

Les seconds reçoivent les fichiers d'habillage et n'ont donc pas à effectuer tout le processus de génération. Le chapitre "Exemples d'habillage PAW", sous-chapitre "Habillage d'un produit IBM : DSMS" s'adresse à eux, apporte des informations spécifiques et indique quelles parties des autres chapitres doivent être lues pour une mise en place correcte de l'habillage fourni.

1. Présentation des postes développeur et utilisateur

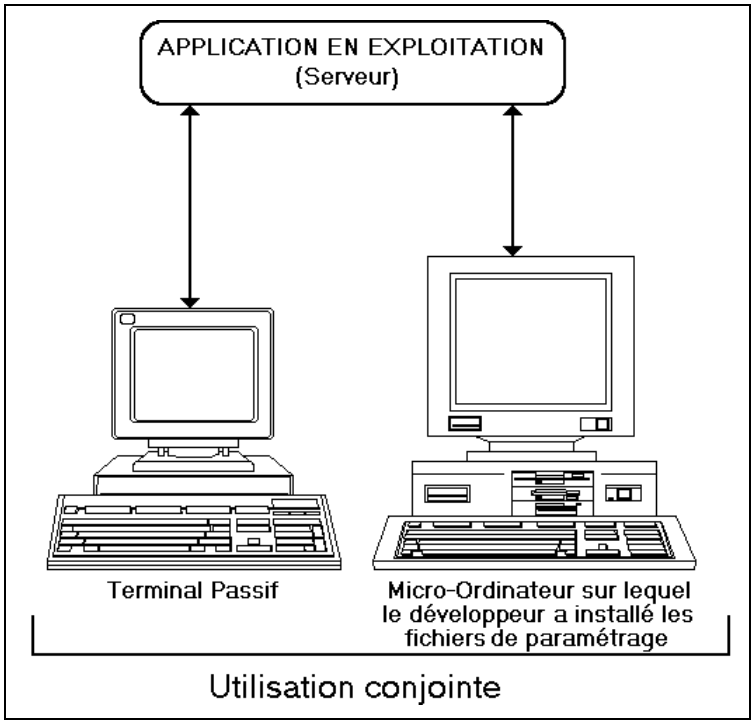
1.1. Poste développeur

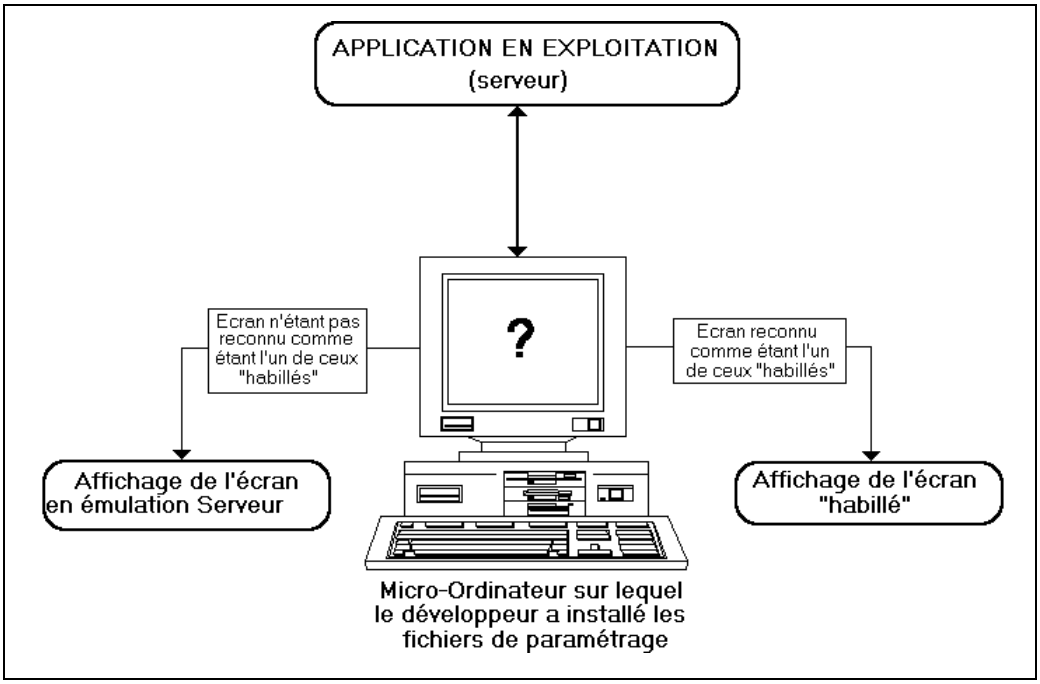
Le module PAWGEN génère le paramétrage d'habillage d'applications écrites avec le module DIALOGUE de PACBASE à partir d'informations extraites de la base. Cette phase de génération est automatisée et demande le minimum d'interventions de la part du développeur.



En ce qui concerne la configuration logicielle et matérielle, il faut vous reporter à la documentation commerciale ou contacter votre support technique.

1.2. Poste utilisateur





Le poste de l'utilisateur est un micro-ordinateur sur lequel le produit PAW va être ou a été installé et duquel on peut accéder au paramétrage d'habillage produit sur le poste du développeur.

Il est important de noter que l'application n'est en rien modifiée. Seule l'interface avec l'utilisateur est nouvelle. En cas de modifications sur le serveur, PAW travaille en mode mixte, le poste de travail passant automatiquement en mode émulation lorsqu'il reçoit un écran dont la version locale d'habillage n'existe pas ou n'est plus à jour.

2. Installation

La première étape consiste donc à installer le poste développeur car c'est à partir de lui que vous pourrez habiller les écrans avant de les transférer sur le poste de l'utilisateur.

Après l'installation du poste développeur, vous pouvez installer le poste utilisateur puis procéder à l'habillage de l'application, ou inversement.

2.1. Procédure d'installation

Pour lancer l'installation d'un poste développeur ou utilisateur, lancer WINDOWS et sélectionner, dans le menu Fichier, le choix Exécuter. Vous pouvez alors choisir d'installer PAW :

- à partir des disquettes. Pour cela, introduire la première disquette d'installation dans le lecteur (A ou B); sélectionner ce lecteur (en tapant A: ou B:) puis taper SETUP.
- à partir d'un serveur sur lequel a été préalablement copié le contenu des deux disquettes d'installation dans deux sous-répertoires différents. Activer le programme SETUP en indiquant son chemin d'accès : ex : X:*chemin_du_disque*\DSK01\SETUP, où
  X = unité du serveur où le contenu des disquettes a été recopié,
  DSK01 = premier sous-répertoire.

La procédure affiche alors une série de boîtes de dialogue qui vous permettent de définir votre environnement de travail.

2.1.1. Caractéristiques communes des boîtes de dialogue

Toutes les boîtes de dialogue contiennent deux boutons-poussoirs :

- un bouton OK qui vous permet de passer à la boîte de dialogue suivante en enregistrant vos choix,
- un bouton Cancel qui vous permet d'interrompre l'installation. Si, dans la boîte de message qui s'affiche, vous cliquez sur 'Oui', vous sortez de la procédure d'installation, et tous les choix sélectionnés jusqu'alors seront perdus. Si vous cliquez sur 'Non', vous reprenez au point où vous étiez quand vous avez appuyé sur Cancel.

A la droite des boîtes de dialogue et de message se trouve l'icône EXIT qui vous permet de sortir de la procédure d'installation, en annulant tous les choix sélectionnés jusque là. Cette icône a donc la même fonctionnalité que le bouton Cancel, mais, contrairement à ce dernier, elle est disponible à tout moment.

2.1.2. Boîte 'Langue d'installation'

Vous devez choisir, entre le français (valeur par défaut) et l'anglais, la langue des libellés des boîtes de dialogue de la procédure d'installation et du compte-rendu d'exécution de PAWGEN.

2.1.3. Boîte 'Langue de l'utilisateur PAW'

Vous devez choisir, entre le français (valeur par défaut), l'anglais et l'espagnol, la langue dans laquelle les libellés et les messages d'erreur s'afficheront dans l'application habillée.

2.1.4. Boîte 'Répertoire ENVIRON.PRM'

Vous devez saisir le chemin complet des exécutables. La valeur par défaut, '**c:\paw\exe**', est modifiable.

Si le répertoire existe déjà (si vous ré-installez PAW sur un PAW déjà existant), la procédure y recherche un éventuel fichier ENVIRON.PRM dans lequel sont regroupés les paramètres d'environnement. Si ce fichier est présent, la procédure d'installation vous proposera alors par défaut les choix correspondant à ceux du fichier ENVIRON.PRM existant. Vous n'aurez donc qu'à valider ces choix.

Le début de ce chemin sera le répertoire racine par défaut proposé dans la boîte de dialogue suivante.

2.1.5. Boîte 'Paramètres d'installation PAW'

2.1.5.1. Cadre 'Type de poste'

Dans le cadre 'Type de poste', vous indiquez si vous installez un poste développeur ou utilisateur.

2.1.5.2. Cadre 'Mode d'apprentissage'

L'apprentissage et la reconnaissance des écrans sont basés sur des algorithmes dont le but est d'associer une DLL à un message en provenance du serveur.

Par défaut, la reconnaissance est fondée sur les critères suivants :

- protection ou non de certaines zones,
- valeurs de certaines zones,
- nombre de zones saisissables et longueur cumulée de ces zones,
- identité du contenu des libellés fixes du message reçu du serveur, et du contenu des libellés fixes de la DLL.

Avec ces critères, l'habillage d'une application modifiant dynamiquement l'argument saisissable/protégé d'une ou de plusieurs zones aboutit à une non-reconnaissance de l'écran modifié. Pour pallier ce problème, plusieurs modes d'apprentissage sont à la disposition du développeur.

Par défaut, ce paramètre est positionné à '1: standard', correspondant au cas étudié ci-dessus.

Positionné à '2: par libellés', ce paramètre base uniquement l'apprentissage et la reconnaissance des écrans sur l'identité du contenu des libellés fixes du message reçu du serveur et de la DLL.

Positionné à '3: personnalisé', ce paramètre reprend les critères de reconnaissance par défaut, en modulant le critère de protection par la prise en compte d'un paramètre (VARPRO). Ce paramètre indique quelles zones peuvent passer de saisissables à protégées, permettant ainsi la reconnaissance d'écrans modifiés dynamiquement. Pour des détails sur ce paramètre, se référer au chapitre "Fonctions avancées", sous-chapitre "Liste de valeurs externes", section "Mise en correspondance et enrichissement des caractéristiques d'une rubrique".

Voici quelques conseils vous permettant de choisir entre le mode 1, 2 et 3 :

- si l'application serveur ne modifie pas dynamiquement les attributs ou si elle ne modifie que très peu de zones, le mode d'apprentissage standard (mode 1) est conseillé. Dans le second cas de figure, il faut décrire dans la base serveur toutes les variantes des écrans concernés par la modification dynamique d'attributs.

- si l'application serveur modifie dynamiquement les attributs de plusieurs zones et si les écrans concernés contiennent des libellés fixes, le mode 2 est conseillé.

- si l'application serveur modifie dynamiquement les attributs de plusieurs zones et si les écrans concernés ne contiennent pas ou peu de libellés fixes, le mode 3 est conseillé. Dans ce dernier cas, ne pas omettre de mettre à jour correctement le fichier de mise en correspondance pour indiquer quelles zones sont susceptibles de voir leur attribut modifié.

Positionné à '4: sans apprentissage', ce paramètre indique qu'aucun apprentissage préalable ne doit avoir lieu. La reconnaissance et l'habillage des écrans se fait grâce au repérage, dans cet écran, d'un libellé fixe prévu par l'utilisateur dans PACBASE, et dont les coordonnées sont mémorisées en ligne 037 du fichier ENVIRON.PRM. Ce libellé, composé de 1 à 8 caractères, donne le nom de la DLL de l'écran à habiller. On trouve ci-après l'exemple du repérage d'un tel libellé:

[,L01,C001,N8,]

Il est ici défini comme localisé en ligne 1 (**L01**), colonne 1 (**C001**), et ayant une longueur de 8 caractères (**N8**).

Si le libellé trouvé dans l'écran est, par exemple, **PB16001O**, PAW ira chercher une DLL de nom **PB16001O.DLL**.

Mais il se peut que l'utilisateur ait plusieurs jeux de DLL, de noms différents, pour des mêmes écrans à habiller. Dans ce cas, il pourra définir (dans le fichier ENVIRON.PRM) des chaînes de caractères à concaténer en début et en fin de la chaîne prévue dans ces écrans. Cette dernière devant alors avoir une longueur inférieure à 8 caractères.

Exemple :

[PB,L02,C072,N4,MA]

Ici, la chaîne à identifier est localisée en ligne 2 (**L02**), colonne 72 (**C072**), et a une longueur de 4 caractères (**N4**).

Si le libellé trouvé dans l'écran est **LO10**, PAW ira chercher une DLL de nom **PBLO10MA.DLL**. Si, dans l'écran suivant de l'application, la chaîne lue est **LO20**, PAW ira chercher la DLL de nom **PBLO20MA.DLL**.

En tout état de cause, l'ensemble *chaîne_avant+chaîne_lue+chaîne_après* ne devra pas avoir une longueur totale supérieure à 8 caractères.

2.1.5.3. Cadre 'Répertoire racine'

Cette information n'est pas obligatoire. Elle représente une facilité de saisie. En effet, si vous saisissez un réperoire racine, en indiquant un chemin absolu (par exemple **C:\PAW**) dans ce cadre, il constituera automatiquement le début du chemin des autres répertoires. Cependant, si vous voulez ne pas faire dépendre tous les répertoires de ce répertoire racine, vous pouvez modifier le chemin de certains répertoires.

Si vous ne saisissez aucun répertoire racine, vous devrez préciser le chemin complet de chaque répertoire.

La valeur proposée par défaut pour le répertoire racine est la première partie du chemin des exécutables indiqué dans la boîte 'Répertoire ENVIRON.PRM'.

2.1.5.4. Cadre 'Unité'

Vous devez choisir, parmi toutes les valeurs proposées, le disque sur lequel PAW sera installé.

2.1.5.5. Cadre 'Espace disque'

Les informations contenues dans ce cadre ne sont pas saisissables. Elles changent en fonction de l'unité choisie.

Sont indiqués, en octets, l'espace disque total et l'espace disponible avant l'installation de PAW. Puis sont indiqués l'espace disque total (identique à celui de la première ligne) et l'espace disponible après l'installation de PAW.

Pour information, la version PAW 1.6 requiert un espace d'environ 1,4 MO. Quant à DSMS habillé, il nécessite un espace d'environ 8 MO.

Vous devez vous-même vérifier que l'espace disponible est suffisant pour tous vos fichiers déjà présents sur le disque et pour ceux que vous comptez installer. Si ce n'est pas le cas, quitter l'installation en appuyant sur le bouton-poussoir Cancel ou l'icône 'EXIT' et faire de la place sur le disque.

2.1.6. Boîte 'Répertoires des composants de PAW'

C'est dans ces répertoires que vont être installés tous les fichiers nécessaires au fonctionnement de PAW.

Le répertoire racine n'est pas saisissable. La valeur affichée (s'il y en a une), provient de la boîte de dialogue précédente.

Par défaut, tous les répertoires proposés commencent par le répertoire racine. Mais vous pouvez indiquer un chemin totalement différent pour certains répertoires.

Toutes les valeurs proposées dans les champs de saisie sont des valeurs par défaut que vous pouvez modifier.

Les sept répertoires sont :

- **Exécutables** : répertoire des fichiers .EXE, des fichiers de paramétrage pour la génération, du fichier ENVIRON.PRM et du fichier de test PAWTEST1.DAT. Il sera appelé CHEMIN-EXE par la suite.
- **Communication** : répertoire des fichiers .PRM et .TAB nécessaires aux différents protocoles de communication.
- **Personnalisation** : à l'issue de l'installation, ce répertoire contient la DLL des icônes du clavier logiciel. A l'issue de la première utilisation, il contient en plus les fichiers des préférences (couleur, etc.) et le paramétrage du clavier.
- **DLL d'habillage** : à l'issue de l'installation, ce répertoire contient l'aide en français, anglais et espagnol (se référer à la section "Fin de l'installation", paragraphe "Poste de l'utilisateur" pour des explications sur la protection de ces fichiers). Une fois l'application habillée installée, il contient en plus les fichiers de paramétrage compilés, nécessaires au fonctionnement de l'application sur le poste de l'utilisateur. Il sera appelé CHEMIN-ECRANS par la suite.
- **DLL listes externes**. A l'issue de l'installation, ce répertoire est vide. Il sera appelé CHEMIN-LIST par la suite.
- **Base du développeur** : à l'issue de l'installation, ce répertoire contient la description des icônes du clavier logiciel et un utilitaire pour regénérer le clavier logiciel. Une fois l'application habillée installée, ce répertoire est la racine de l'arborescence dans laquelle sont produits les fichier DLL et HLP de l'application. Il sera appelé CHEMIN-BASE par la suite. Pour un poste utilisateur, ce répertoire reste vide.
- **Scripts**: uniquement sous WINDOWS. Répertoire des fichiers de scripts et des fichiers associés (CONNEX.PRM...).

Le bouton << situé en bas à gauche vous permet de revenir à la boîte de dialogue précédente, pour corriger par exemple le nom du répertoire racine.

2.1.7. Boîte 'Répertoire de Windows'

En fin d'installation, l'icône de PAW est créée dans un dossier du même nom. Il est donc nécessaire que l'utilisateur donne à l'installateur le nom du répertoire de Windows dans lequel doit figurer PAW.

Par défaut, le répertoire présenté est '**c:\windows**', mais il est possible de le modifier en fonction du nom réel que vous avez choisi sur votre poste.

2.1.8. Boîte 'Protocole de Communication'

Vous choisissez dans cette boîte le protocole de communication sous lequel le serveur fonctionne (ex : IBM/BULL 3270).

2.1.9. Boîte 'Variante de Protocole'

Vous choisissez dans cette boîte un type de communication, dans le cas où le protocole en autorise plusieurs.

Le bouton << situé en bas à gauche vous permet de revenir à la boîte de dialogue précédente, pour corriger le protocole de communication.

2.1.10. Boîte 'Réinstallation de PAW sur une version précédente'

Si le répertoire racine défini en début d'installation existe déjà sur le poste, et uniquement dans ce cas, il vous est demandé de choisir entre trois options :

- soit recréer le répertoire (et ses sous-répertoires): l'ancienne version de PAW sera donc détuite.

- soit installer la nouvelle version 'par-dessus' la précédente: on conserve l'arborescence préexistante, on y installe les composants de PAW tout en préservant les éventuels autres fichiers qui s'y trouveraient.

- soit revenir en arrière dans les étapes de l'installation pour changer de répertoire.

2.1.11. Fin de l'installation

La procédure d'installation commence à installer les sous-répertoires, vous demande d'insérer la deuxième disquette puis vous propose un nom pour la fenêtre PAW sous WINDOWS. C'est le nom qui s'affichera dans le groupe qui inclura l'icône PAW.

Lorsque l'installation est terminée, une boîte de message 'Fin de l'installation' s'affiche.

2.1.11.1. Poste du développeur

La procédure d'installation met en place sur le poste du développeur les différents éléments nécessaires à l'habillage de l'application. Ces éléments sont :

- un répertoire (désigné ici par CHEMIN-EXE) contenant les modules PAWGEN (qui génère les sources des écrans à habiller) et PAWLIS (utilisé optionnellement pour générer des listes de valeurs externes) et leurs fichiers de paramètres, ainsi que le module PAW pour les tests.
- un répertoire (désigné ici par CHEMIN-BASE) qui sert de racine aux ensembles produits par PAWGEN et PAWLIS.

2.1.11.2. Poste de l'utilisateur

Lors de l'installation de PAW sur le poste de l'utilisateur, trois fichiers d'aide sont automatiquement copiés dans le répertoire lié au paramètre CHEMIN-ECRANS. Il s'agit des fichiers PW_HLPEN.HLP (aide en anglais), PW_HLPES.HLP (aide en espagnol) et PW_HLPFR.HLP (aide en français). Comme c'est également dans ce répertoire que sont copiés les fichiers d'habillage réalisés sur le poste du développeur et que ces fichiers doivent être de temps à autre mis à jour (destruction des vieilles versions et copie de nouvelles), il faut pouvoir protéger ces trois fichiers d'aide (ou un seul si vous ne voulez garder qu'une seule langue) contre toute destruction accidentelle.

Pour protéger par exemple le fichier d'aide français contre toute destruction ou altération, utiliser la commande :

ATTRIB +R PW_HLPFR.HLP

Pour lever la protection, utiliser la commande :

ATTRIB -R PW_HLPFR.HLP

2.2. Personnalisation du poste installé

2.2.1. Fichier ENVIRON.PRM

Le fichier ENVIRON.PRM est créé par la procédure d'installation dans le répertoire de chemin d'accès CHEMIN-EXE. Le développeur peut le modifier en fonction de ses besoins en utilisant un éditeur de texte de son choix. Il doit respecter la syntaxe décrite ci-après.

Les lignes de ENVIRON.PRM ont la structure suivante :

xxx commentaire [**paramètre**]

où

- **xxx** est un nombre occupant les 3 premiers caractères de la ligne.
- **commentaire** est un commentaire facultatif.
- **paramètre** est la valeur du paramètre écrite entre crochets.

Les lignes n'ayant pas cette structure sont ignorées. On peut donc introduire des commentaires dans ce fichier.

Les lignes possédant un numéro inférieur à 20 sont obligatoires. Les autres lignes (exceptée la ligne 27) correspondent à un paramétrage facultatif dont l'absence n'entrave pas le bon fonctionnement de PAW.

001	Version et Langue	[1.6 FR FR]
002	Carte de communication	[GSDLL32 GSPC32W]
003	Chemin des EXE	[c:\paw160\exe]
004	Chemin de l'utilisateur	[c:\paw160\ecr]
005	Chemin des listes de valeurs	[c:\paw160\lis]
006	Chemin du développeur	[c:\paw160\dev]
007	Système cible	[DOS]
020	Période de lecture d'écran	[700]
021	Tolérances DIALOGUE	[0]
022	DLLs avec noms externes	[0]
023	Mode d'apprentissage	[1]
024	Type d'utilisateur	[d]
027	Chemin des utilitaires GSCOM	[c:\paw160\com]
028	Fichiers de paramétrage	[c:\paw160\per]
029	DLL des icônes	[ICOPAW]
030	Chemin des scripts	[c:\paw160\scr]
031	Affichage fenêtre des scripts	[0]
032	Nom de l'application DDE	[PAW]
033	Nom de l'application habillée	[PAW]
037	Identification de la DLL	[,L01,C001,N8,]

- **Version et Langue (001) :** code composé d'un numéro de version, suivi de deux codes langue sur deux caractères. Le premier code langue représente la langue de l'utilisateur et le second celle de l'installation. Vous les avez toutes deux sélectionnées dans les deux premières boîtes de dialogue de la procédure d'installation. Les valeurs possibles sont FR pour le français, EN pour l'anglais et ES pour l'espagnol.

- **Carte de communication (002) :** nom du gestionnaire de communication (l'extension .EXE est sous-entendue), suivi du nom du fichier de paramètres qui lui est associé (l'extension .PRM est sous-entendue).

- **Chemin des EXE (003) :** chemin du répertoire dans lequel sont placés les programmes spécifiques à PAW. Sur un poste développeur se trouvent les programmes PAW.EXE, PAWGEN.EXE et PAWLIS.EXE car le développeur a besoin de ces trois programmes pour réaliser l'habillage des écrans. En revanche, sur un poste utilisateur, seul PAW.EXE est présent. Ce paramètre est le CHEMIN-EXE de la procédure d'installation.

- **Chemin de l'utilisateur (004) :** chemin du répertoire dans lequel sont placés, dès l'installation, les fichiers d'aide générés automatiquement, et, une fois l'application habillée installée, les fichiers de paramétrage compilés, nécessaires au fonctionnement de l'application sur le poste de l'utilisateur. Ce paramètre est le CHEMIN-ECRANS de la procédure d'installation.

- **Chemin des listes de valeurs (005) :** chemin du répertoire dans lequel sont copiés les fichiers compilés contenant les listes de valeurs externes. Ce paramètre est le CHEMIN-LIST de la procédure d'installation. CHEMIN-LIST doit être distinct de CHEMIN-ECRANS**.**

- **Chemin du développeur (006) :** début du chemin du répertoire dans lequel sont placés les fichiers résultant des phases de génération et de compilation. Le chemin complet est constitué par ajout des répertoires SESS (numéro de la session) et BIB (nom de la bibliothèque contenant les écrans traités). Ce paramètre est le CHEMIN-BASE de la procédure d'installation.

- **Système cible (007) :** DOS. système sur lequel fonctionnera l'application habillée. Il conditionne la génération par PAWGEN de fichiers d'habillage au format MS-WINDOWS. Ce paramètre est le SYSTEME de la procédure d'installation.

- **Période de lecture d'écran (020) :** Ce paramètre, facultatif, permet d'exprimer en millisecondes le délai qui sépare deux interrogations de la carte de communication en vue d'effectuer un contrôle de transmission. Cette valeur doit être, de préférence, supérieure à 20 millisecondes. La valeur 0 annule le contrôle de la carte (équivalent à la suppression de la ligne).

- **Tolérances DIALOGUE (021) :** 0 ou 1 (paramètre facultatif). La valeur 0 génère un paramétrage totalement conforme aux lignes de descriptions PACBASE des écrans.
  Avec la valeur 1, PAWGEN effectue les mêmes ajustements que le module Dialogue de PACBASE.
  Lorsqu'un écran comporte une zone commençant en ligne 1 colonne 1, Dialogue de PACBASE décale cette zone en ligne 1 colonne 2, pour pouvoir intercaler un attribut.
  Lorsqu'un écran comporte une zone finissant sur la dernière colonne d'une ligne alors que la première zone de la ligne suivant commence en colonne 1, Dialogue de PACBASE décale automatiquement la deuxième pour insérer un attribut.
  Avec la valeur 0, il existe des risques de déphasage entre la description locale et la grille générée par le serveur. En cas de problème d'affichage, consulter le chapitre "Gestion des erreurs", sous-chapitre "Erreur à l'utilisation".

- **DLLs avec noms externes (022) :** 0 ou 1. La valeur 0 (défaut) provoque la génération de fichiers de paramétrage ayant comme radical le code des écrans auxquels ils correspondent.
  Avec la valeur 1, c'est le code externe (MAP) qui est utilisé comme radical, ce qui permet de placer dans un même répertoire des fichiers de paramétrage en plusieurs langues correspondant à un même écran serveur (les MAPS ne portant pas le même nom d'une langue à l'autre).
  Ce paramètre est obligatoire et doit être positionné à 1 dans le cadre du portage d'une application MICROFOCUS vers une application habillée par PAW.

- **Mode d'apprentissage (023) :** 1, 2**,** 3 4. A l'aide de ce paramètre, il est possible de choisir l'algorithme de reconnaissance des écrans.
  Ceci permet notamment de pallier les problèmes posés par les programmes modifiant dynamiquement les attributs des zones de saisie (risque de non-reconnaissance des écrans modifiés).
  Pour obtenir de plus amples renseignements à ce sujet, consulter le sous-chapitre "Procédure d'installation", section "Paramètres d'installation PAW", paragraphe "Cadre Mode d'apprentissage".

- **Type d'utilisateur (024) :** D ou U. Cette ligne indique si le poste fonctionne comme un poste développeur ou comme un poste utilisateur.

- **Chemin des utilitaires GSCOM (027) :** Ce répertoire permet de stocker les fichiers de communication.

- **Fichiers de paramétrage (028) :** Ce répertoire permet de stocker les préférences et les paramétrages du clavier effectués dans un souci de personnalisation.

- **DLL des icônes (029) :** Ce code est le nom de la DLL contenant les icônes utilisées dans le clavier logiciel.

- **Chemin des scripts (030) :** Cette ligne correspond au chemin du répertoire dans lequel sont placés les fichiers de scripts nécessaires à l'exécution du programme.

- **Affichage fenêtre des scripts (031) :** 0, 1 ou 2. Cette ligne permet d'afficher ou non le déroulement du script pour mise au point. 0 : affichage du titre du script. 1 : affichage du déroulement complet du script. 2 : aucun affichage.

- **Nom de l'application DDE (032) :** par défaut, la valeur est 'PAW'. Il est possible de la modifier en fonction des besoins.

- **Nom de l'application habillée (033) :** par défaut, la valeur est 'PAW'. Il est possible de la modifier en fonction des besoins.

- **Version de l'application habillée (036) :** mémorisée dans SCREENS.LRN. Elle est affichée dans l'aide sur PAW. ATTENTION: si ce paramètre est modifié, il faut détruire SCREENS.LRN pour que la modification soit prise en compte.

- **Identification de la DLL (037) :** coordonnées de la chaîne de caractères déterminant de nom de la DLL d'habillage (mode d'apprentissage 4).

2.2.2. Fichier PAWLIB.PRM

Le fichier PAWLIB.PRM est créé par la procédure d'installation dans le répertoire de chemin d'accès CHEMIN-EXE. Son contenu permet de traduire dans la langue de l'utilisateur certains des libellés qui s'afficheront dans les menus ACTION et ENCHAINEMENT de la barre de menus de son poste, et certains libellés généraux liés à l'aide.

Le tableau ci-dessous correspond à la version d'installation du fichier PAWLIB.PRM.

Les lignes de ce fichier ont la structure suivante :

xxx commentaire [**libellé**]

où

- **xxx** est un nombre occupant les 3 premiers caractères de la ligne.
- **commentaire** est un commentaire facultatif.
- **libellé** contient le libellé à utiliser. Il doit être écrit entre crochets. Sa longueur ne doit pas excéder 36 caractères.

Les lignes n'ayant pas cette structure sont ignorées. On peut donc introduire des commentaires dans ce fichier.

010	CMVT__	[Pas de mise à jour]
011	CMVT__C	[Création]
012	CMVT__M	[Modification]
013	CMVT__A	[Annulation]
014	CMVT__X	[Créat.ou modification selon l'utilisateur]
015	CMVT__Y	[Mouvement 5]
016	CMVT__Z	[Mouvement 6]
020	OPER__P	[Même écran, même contenu]
021	OPER__A	[Même écran, contenu au choix de l'utilisateur]
022	OPER__S	[Même écran, suite de liste]
023	OPER__M	[Mise à jour]
024	OPER__O	[Autre écran]
025	OPER__E	[Fin de conversation]
040		[Pas d'aide étendue pour cet écran]
041		[Aide sur l'écran :]

- de 10 à 16 :
  libellés explicatifs des valeurs possibles d'un code mouvement. Ces libellés mettent à jour le menu ***Action***.

- de 20 à 25 :
  libellés explicatifs des valeurs possibles d'un code opération. Ces libellés mettent à jour le menu ***Enchaînement***.

- 40 et 41 : aide

- 40 : libellé apparaissant quand il n'existe pas d'aide étendue sur un écran (si la fonction souffleur sur l'écran serveur habillé n'a pas été remplie).
- 41 : préfixe de l'aide apparaissant en début de titre des panneaux d'aide sur l'écran habillé.

2.2.3. Fichier PAWMAK.PRM

Le fichier PAWMAK.PRM est créé par la procédure d'installation dans le répertoire de chemin d'accès CHEMIN-EXE. Il permet de paramétrer la compilation et l'édition de liens des fichiers produits par PAWGEN dans le répertoire CHEMIN-BASE\SESSION\BIBL et/ou PAWLIS dans le répertoire CHEMIN-BASE\LISTE**.**

Les fichiers PAWMAK.MC6, PAWMAK.MC7, PAWMAK.MC8 et PAWMAK.BL4, permettant de paramétrer la compilation et l'édition de liens pour MICROSOFT C6, C7, C8 et BORLAND C4, sont copiés lors de l'installation du poste développeur dans le répertoire des exécutables. PAWMAK.PRM n'est pas installé s'il existe déjà dans le répertoire afin que le développeur puisse conserver les options qu'il a mises au point. Sinon, PAWMAK.PRM est identique à PAWMAK.MC8.

La structure du fichier PAWMAK.PRM suit l'exemple suivant :

```
****   PAWMAK DOS Microsoft C 8.00 Français

[COMP]

FORMAT=cl -c/ALw -Gsw -Ot -Zpe -W2 -FPa -Tc <SRCPAW>.C


[LINK]

FORMAT=link <OBJPAW>, <DLLPAW>.DLL /align:16, NUL, LIBW.LIB
LDLLCEW.LIB /nod /NOE, <DLLPAW>.DEF

FORMAT=rc <DLLPAW>.DLL


[HELP]

FORMAT=HC31<DLLPAW>.HPJ


[UTIL]

COMMAND=NMAKE
```

Les trois étapes de compilation, d'édition de liens et de génération sont associées chacune à une section dont le nom est écrit entre crochets ([COMP]). Dans chaque section, une ou plusieurs lignes décrivent la syntaxe des ordres.

Il existe trois mots réservés :

- <DLLPAW> : code de la DLL et des fichiers d'aide.
- <SCRPAW> : code des fichiers sources et .OBJ (fichiers intermédiaires).
- <OBJPAW> : code des fichiers .OBJ dans l'ordre d'édition des liens.

Ils sont remplacés par les codes appropriés au moment de la génération.

3. Habillage d'une application

La génération des fichiers de paramétrage permettant l'habillage d'applications écrites avec le module DIALOGUE de PACBASE est effectuée par le module PAWGEN qui traite des données extraites de la base PACBASE. Cette phase d'habillage est automatisée et demande le minimum d'intervention de la part du développeur. Elle produit pour chaque écran un ensemble de fichiers à compiler.

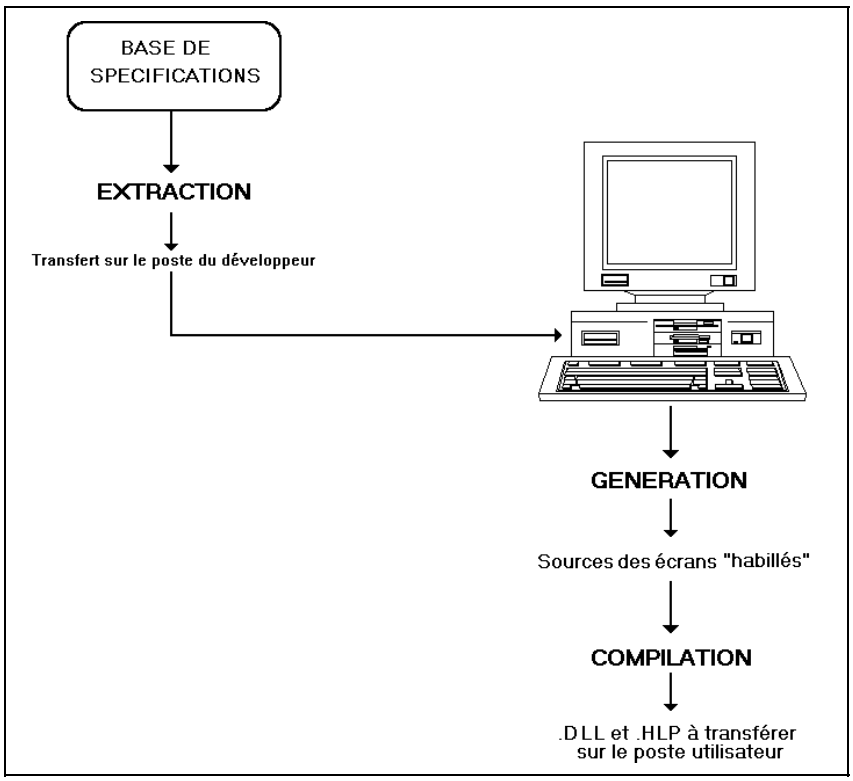
Après cette compilation, le développeur obtient :

- un fichier contenant un masque de l'écran, d'extension .DLL.
- un fichier d'aide, d'extension .HLP, contenant les informations documentaires sur l'écran et les rubriques qu'il utilise.

Ces fichiers sont ensuite installés sur le poste de l'utilisateur.

L'habillage d'une application comporte les tâches suivantes :

- extraction du site central de l'ensemble des informations relatives au dialogue dont les écrans doivent être "habillés",
- transfert sur le poste développeur,
- tests sur les étapes de génération et de compilation,
- génération des sources pour chacun des écrans à "habiller", avec des extensions .C et .RTF (DOS MS-WINDOWS),
- compilation des fichiers obtenus,
- mise en exploitation sur les postes des utilisateurs.



3.1. Extraction des Ecrans sur serveur

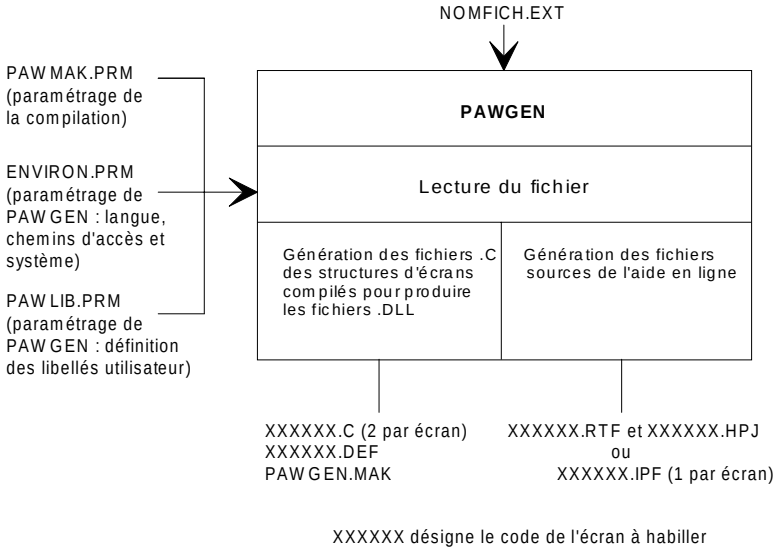
Le fichier qui contient les descriptions des écrans à habiller est produit par la procédure PACBASE GPRT (GEO option C4) sur le serveur. En sortie de la procédure, l'utilisateur obtient le fichier PAC7GT (ou GT pour le système GCOS8) dont chaque enregistrement contient au maximum 180 caractères.

Pour de plus amples informations au sujet de la procédure GPRT, consulter les manuels d'utilisation et d'exploitation PACBASE.

3.2. Transfert du fichier d'extraction sur micro-ordinateur

La descente du fichier PAC7GT ou GT sur micro-ordinateur est à la charge du développeur. L'utilitaire de transfert doit être paramétré de telle sorte que les caractères spéciaux (accentués notamment) soient bien conservés. Nous désignerons par NOMFICH.EXT le fichier local résultant du transfert.

3.3. Génération des sources des écrans à "habiller"



A partir de NOMFICH.EXT, PAWGEN permet d'obtenir l'ensemble des sources à compiler pour chacun des écrans considérés.

Pour lancer la procédure, saisir la commande suivante :

PAWGEN NOMFICH.EXT

où NOMFICH.EXT désigne le nom complet du fichier obtenu par la procédure GPRT et descendu en local. Ce nom doit contenir les indications suffisantes concernant le chemin d'accès.

PAWGEN produit des fichiers sources .C et des fichiers sources d'aide (.RTF pour MS-WINDOWS).

Ces fichiers sont créés dans un répertoire dont le nom est constitué du CHEMIN-BASE\SESSION\BIB_BIBL où :

- CHEMIN-BASE est le chemin du développeur (ligne 006 du fichier ENVIRON.PRM),

- SESSION est le code de la session d’extraction,
- BIBL_bib est le code de la bibliothèque d'où les mouvements ont été extraits.

Dans le cas de la session courante, le répertoire SESSION est toujours le répertoire 9999H**.** Les répertoires SESSION et BIBL sont automatiquement créés à partir du répertoire CHEMIN-BASE. Le paramétrage est introduit au moyen des fichiers ENVIRON.PRM, PAWLIB.PRM et PAWMAK.PRM étudiés dans le chapitre "Installation".

Un compte rendu de la procédure de génération est affiché à l'écran. Le développeur peut le rediriger sur un fichier texte consultable a posteriori et vérifier qu'il ne contient pas de message d'erreur. Il s'y trouve :

- des statistiques sur le déroulement de PAWGEN.
- des messages d'anomalies (warning ***).
- des messages d'erreurs (error *1* et error *2*).

  Une erreur 2 signale un problème grave (erreur système, absence d'un fichier de paramétrage...) qui provoque l'arrêt du traitement.

Une liste des erreurs est donnée dans le chapitre "Gestion des erreurs", sous-chapitre "Erreurs de génération".

3.4. Compilation

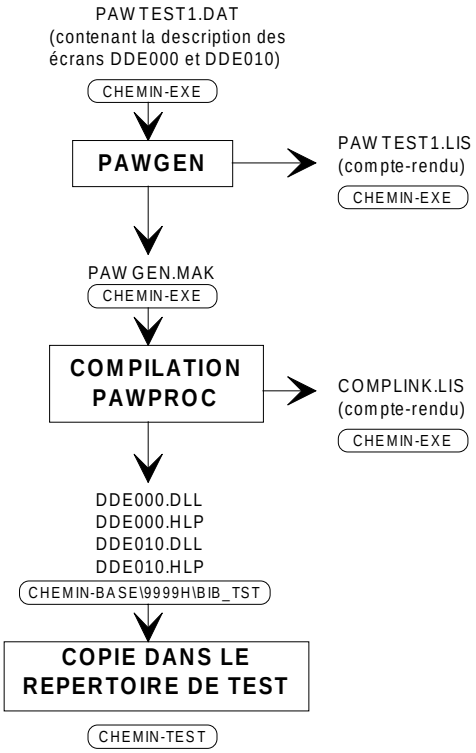
Vous lancez les compilations par la commande PAWPROC.

Les compilations sont :

- compilation des fichiers .C suivie de l'édition des liens pour produire des .DLL.
- compilation des fichiers d'aide (.RTF) à l'aide du compilateur HELP COMPILER**.** Les fichiers définitifs sont écrits dans un répertoire dont le nom est constitué du chemin d'accès du développeur (ligne 005 du fichier ENVIRON.PRM) et des noms de sessions et de bibliothèques.

Le compte-rendu des compilations s'affiche à l'écran. Vérifier qu'il ne contient pas de messages d'erreur.

3.5. Tests sur la génération et la compilation



Afin que le développeur puisse effectuer des tests, le fichier PAWTEST1.DAT est automatiquement copié lors de l'installation dans le répertoire correspondant au paramètre CHEMIN-EXE. Ce fichier contient le descriptif de deux écrans, DDE000 et DDE010, tels qu'ils auraient pu être obtenus par le développeur par une extraction du site central. Lors du test, le fichier PAWTEST1.DAT va subir les différentes phases de la procédure d'habillage, permettant de déceler toute erreur d'installation ou de paramétrage de PAW.

- Génération du paramétrage

Se placer dans le répertoire lié au paramètre d'installation CHEMIN-EXE par l'intermédiaire de la commande CD.

Exécuter ensuite PAWGEN.EXE au moyen de la commande :

PAWGEN PAWTEST1.DAT > PAWTEST1.LIS

Le compte-rendu de génération est écrit dans le fichier PAWTEST1.LIS, consultable grâce à n'importe quel éditeur standard.

- Contrôle de la génération

Editer le fichier de compte-rendu PAWTEST1.LIS. Les lignes d'erreurs débutent par **error* 1* ou **error* 2*. Le compte-rendu ne doit en contenir aucune.

- Compilation et édition des liens.

Lancer la commande PAWPROC.

Si la commande échoue et que le message d'erreur "Exécutable non trouvé" **(**SYS1041**)** apparaît, cela signifie que le système n'a pas trouvé le programme dont l'exécution a été demandée, à savoir le compilateur ou l'éditeur de liens.

Dans ce cas, vérifier que :

- le programme appelé est correctement installé sur le poste du développeur ou sur le serveur auquel le poste est connecté.
- le programme se trouve soit dans le répertoire courant, soit dans un des répertoires mentionnés dans la variable d'environnement PATH.

  Pour connaître le contenu de cette variable, utiliser la commande PATH.

  Pour modifier le PATH, mettre le fichier d'initialisation de l'environnement à jour (AUTOEXEC.BAT pour DOS) puis relancer la machine pour que cette modification soit prise en compte.
- que les noms des programmes appelés (par défaut CL.EXE et LINK.EXE) sont bien ceux du compilateur utilisé. Si tel n'est pas le cas, il faudra modifier le fichier PAWMAK.PRM.

• Contrôle de la compilation et de l'édition des liens.

Le répertoire CHEMIN-BASE\9999H\BIB_TST doit contenir 2 fichiers d'extension DLL et 2 fichiers d'extension HLP. Si ce n'est pas le cas, l'origine en est :

- erreurs de compilation :

  Voir le chapitre "Gestion des erreurs".
- erreurs d'édition des liens :

  Il manque des bibliothèques de modules. Vérifier qu'elles sont installées et le cas échéant qu'elles sont mentionnées dans la variable LIBPATH d'environnement. Pour cela, taper la commande SET.

  Si l'environnement est incorrect, mettre à jour les fichiers d'environnement, puis relancer la machine.

• Copie du résultat dans le répertoire de test

La copie s'effectue au moyen de la commande COPY.

Si le développeur désire au préalable détruire tous les anciens fichiers présents dans ce répertoire par la commande DEL ***.***, il doit auparavant protéger le ou les fichiers PW_HLPEN.HLP (aide en anglais), PW_HLPES.HLP (aide en espagnol) et PW_HLPFR.HLP (aide en français), selon son choix, qui se trouvent également dans ce répertoire (automatiquement copiés lors de l'installation).

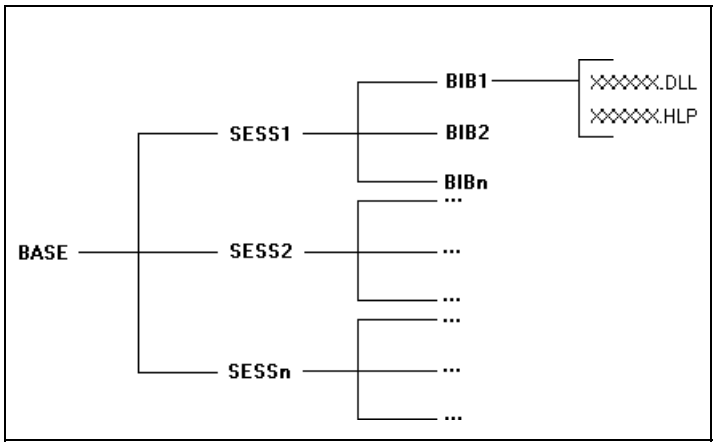
Pour protéger par exemple le fichier d'aide français contre toute destruction ou altération, utiliser la commande :

```
ATTRIB +R PW_HLPFR.HLP
```

Pour lever la protection, utiliser la commande :

```
ATTRIB -R PW_HLPFR.HLP
```

3.6. Mise en exploitation



Le résultat de la génération des sources des écrans à habiller et de leur compilation est la production d'un ensemble de fichiers compilés pour chaque groupe d'écrans appartenant à un couple SESSION\BIB. Ces fichiers sont écrits dans le répertoire CHEMIN-BASE\SESSION\BIB**.**

Si la session d'origine est la session courante, le répertoire SESSION est toujours nommé 9999H.

Pour tester le paramétrage produit, le développeur copie l'ensemble des fichiers sur son répertoire de test (CHEMIN-ECRANS) par la commande XFERPROD, fabriquée par le générateur.

Le transfert sur le poste de l'utilisateur s'effectue par une copie du paramétrage produit sur le répertoire CHEMIN-ECRANS du poste de l'utilisateur.

4. Gestion des erreurs

4.1. Erreurs à l'installation

La procédure d'installation détecte et signale les erreurs liées aux paramétrage de PAW. Elle détecte également les erreurs liées à un mauvais paramétrage de l'environnement système. Ces dernières peuvent interrompre le déroulement de l'installation.

En cas de saturation de l'environnement système, il est possible de modifier le fichier CONFIG.SYS pour augmenter la taille de l'environnement, ou pour désactiver certains programmes résidents "gourmands" en mémoire.

Les modifications du fichier CONFIG.SYS ne prennent effet que lorsque le système est relancé.

4.2. Erreurs de génération

Ce sous-chapitre décrit l'ensemble des messages d'erreurs qui peuvent apparaître à l'écran, suite à l'utilisation de PAWGEN ou de PAWLIS. Seules les zones imprimées en gras correspondent aux messages effectivement affichés, le reste du texte expliquant ces derniers.

NUMERO	**LIBELLE DE L'ERREUR**
ANOMALIES DETECTEES DANS ENVIRON.PRM (error* 2*) :**	
10001	Fichier absent
10002	Ligne incorrect
10003	Ligne absente
10004	Ligne en double
10005	Longueur du paramètre incorrecte
10006	Chemin incorrect
10007	Le paramètre doit avoir 0 ou 1
10008	Le paramètre doit être numérique
10009	Version non numérique
10010	Code langue inconnu
10011	Code système inconnu
10012	Code incorrect (ASCII ou ANSI)
10013	Le chemin de l’utilisateur et celui des listes sont égaux
10014	Valeur du paramètre inconnue

10043 **Fichier PAWLIB.PRM absent ou inutilisable**
****error* 2*** Le fichier de paramétrage n'est pas sur le même répertoire que le programme PAWGEN.

10045 **Fichier PAWMAK.PRM absent ou inutilisable**
****error* 2*** Le fichier de paramétrage n'est pas sur le même répertoire que le programme PAWGEN.

10047 **Fichier des listes de valeurs absent ou inutilisable**
****error* 2*** Le fichier de mise en correspondance (intitulé **RUB-LIST.DAT** dans cet ouvrage) n'existe pas ou ne correspond pas au nom indiqué lors de la génération des fichiers d'habillage.

10050 **Fichier non traité.**
Le fichier n'a pas été traité. Aucun paramétrage n'a été produit pour l'habillage. Ceci est la conséquence d'une erreur 10045, 10055, 10105, 10300 ou 10310.

10055 **Fichier introuvable.**
Le fichier extrait de la base Pacbase n'a pu être lu. Soit il n'existe pas, soit le chemin donné est faux.

10100 **Code inconnu**
****error* 2*** Erreur système lors de l'habillage. Appeler la maintenance.

10105 **Fichier non trié**
****error* 2*** Le fichier n'est pas trié en ASCII. L'habillage est impossible.

10110 **Erreur de séquence.**
****warning**** L'enregistrement dont la clef suit n'est pas traité par PAWGEN. Ce message est normal et peut être ignoré si l'habillage se déroule sans erreur grave.

10180 **Pb d'insertion de zone**
10185 **Pb d'insertion d'hyper**
10190 **Pb de complétion de paragraphe**
****error* 2*** Erreurs système lors de la construction d'un fichier d'aide. Appeler la maintenance.

10200 **Champ de longueur nulle dans l'enreg. :**
****warning**** Une rubrique a été décrite avec une longueur nulle. Le champ correspondant ne sera pas écrit dans le fichier.DLL.

10210 **L'écran est vide**
****error* 1*** L'écran ne contient ni zones logiques ni zones physiques. Aucun fichier d'habillage ne sera fourni pour cet écran.

10300 **Problème d'accés au fichier ENVIRON.PRM**
****error* 2*** Le fichier ENVIRON.PRM n'a pu être ouvert pour lecture. Vérifier qu'il n'est pas utilisé par une autre application.

20070 **Le fichier.. n'a pas le bon format.**
****error* 1*** Erreur système lors du traitement d'un écran. Les fichiers le concernant ne seront pas produits.

20080 **Echec à l'ouverture (mode..) du fichier..**
****error* 1*** Erreur système lors du traitement d'un écran. Les fichiers le concernant ne seront pas produits. Vérifier le fichier indiqué.

20400 **Chevauchement de 2 zones à l'écran :**
****error* 1*** Deux rubriques se chevauchent dans un écran. Les fichiers le concernant ne seront pas produits. Corriger dans la base PACBASE.

20430 **La zone logique.. a une longueur nulle. Suppression du champ correspondant Lig =.. Col =..**
****warning**** La zone en question est supprimée de l'écran. Les fichiers d'habillage sont écrits. Pour savoir de quelle zone il s'agit, se reporter au fichier .C correspondant. Les zones logiques sont indiquées avec leurs numéros.

20440 **Débordement de ligne. Zone.. : Col =.. Long =.. Max =..**
20450 **Débordement d'écran. Zone.. : Lig =.. (Max =..) Col =.. (Max =..)**

****error* 1***	Une rubrique est décrite de telle façon qu'elle déborde de la ligne ou de l'écran. Les fichiers d'habillage de l'écran ne seront pas produits.
20460	**La première zone de l'écran commnence en ligne 1 et colonne 1**
****warning****	Si la ligne 021 du fichier **ENVIRON.PRM** est paramétrée avec 1, le message 20462 est lié au message 20460, indiquant une mise à jour de la description de l'écran pour être en accord avec le module **Dialogue de PACBASE**. Dans le cas contraire, c'est le message 20474 qui est lié au 20460, indiquant un risque de problème.
20462	Décalage en ligne 1 colonne 2
20470	**Pas d'intervalle entre 2 zones à l'écran situées sur 2 lignes consécutives**
****warning****	Si la ligne 021 du fichier **ENVIRON.PRM** est paramétrée avec 1, le message 20472 est lié au message 20470, indiquant une mise à jour de la description de l'écran pour être en accord avec le module **Dialogue de PACBASE**. Dans le cas contraire, c'est le message 20474 qui est lié au 20470, indiquant un risque de problème.
20472	Raccourcissement de la première zone.
20474	L'écran risque de n'être pas reconnu.

4.3. Erreurs de compilation

Si la procédure de compilation et d'édition des liens s'arrête de façon anormale, ceci peut être dû à un manque d'espace mémoire : le système affiche alors un message du type "Out of Heap Space" ou "Out of Far Heap Space".

Suivant le système et le compilateur (se référer à la documentation de celui-ci) certaines options permettent une meilleure utilisation de l'espace mémoire (voir les options comme /B1 C1LEXE etc...).

Si les problèmes persistent, il faut réduire le nombre de programmes résidents qui fonctionnent pendant l'exécution de la procédure. Pour cela modifier le fichier CONFIG.SYS et relancer la machine.

Attention : certains des programmes inactivés pour la phase génération et compilation peuvent être indispensables lors de la phase de test. Il faut donc prévoir de les réactiver lorsque le paramétrage d'habillage a été fabriqué.

4.4. Erreurs à l'utilisation

Il peut arriver que des écrans que le développeur s'attend à voir habillés par PAW s'affichent en émulation.

D'autre part il est possible que dans des cas limites, PAW affiche un écran serveur en utilisant le fichier d'habillage d'un autre s'ils sont très proches.

Ce document permet de contrôler quelques points importants influant sur la reconnaissance, et d'apporter des solutions dans la plupart des cas.

- Ecran non reconnu :

Vérifier les points suivants :

- L'extraction centrale a-t-elle été effectuée ?
- A-t-elle bien été effectuée sur la version de l'application qui est en exploitation (celle à laquelle vous accédez via PAW) ?
- Le transfert sur le poste du développeur a-t-il été correctement effectué ?
- Le logiciel de transfert est-il paramétré de façon à conserver en particulier les caractères accentués (TRES IMPORTANT) ?
- La génération locale (PAWGEN) a-t-elle été lancée ?
- La génération locale s'est-elle déroulée sur le bon fichier (celui qui a été transféré) ?
- La génération s'est-elle déroulée sans erreur ?
- La compilation a-t-elle été lancée ?
- La compilation s'est-elle déroulée sans erreur ?
- La mise en exploitation locale (copie des fichiers dans le répertoire CHEMIN_PARM a-t-elle été faite ?
- Le fichier .DLL de l'écran non reconnu, se trouvant dans le répertoire de test pour le développeur (regarder la mire de l'application PAW ou bien le fichier ENVIRON.PRM), est-il bien de la même taille et date que celui du répertoire BASE\SESSION\BIB où il a été produit par la compilation ?
- Génère-t-on plusieurs fichiers .DLL de même nom? Chaque fichier .DLL doit avoir un nom distinct et ne peut habiller qu'un seul écran. Un fichier .DLL ayant le même nom qu'un autre fichier .DLL écrasera ce fichier qui ne pourra donc plus habiller l'écran associé.
- L'écran non reconnu comporte une zone commençant en ligne 1 colonne 1.

  Dans ce cas, PACBASE décale cette zone en ligne 1 colonne 2, pour pouvoir intercaler un attribut. La grille généréee n'est donc pas en phase avec la description utilisée par PAW. Pour résoudre ce problème, il faut redécrire l'écran avec la zone incriminée en ligne 1 colonne 2. Ceci n'entraine pas de modifications au niveau de l'application, puisque les grilles produites par PACBASE sont inchangées. Il est également possible d'introduire la ligne 021 dans le fichier ENVIRON.PRM pour que dans un tel cas, la mise à jour de la description de l'écran s'effectue de manière automatique.
- L'écran non reconnu comporte une zone finissant sur la dernière colonne d'une ligne alors que la première zone de la ligne suivante commence en colonne 1.

  PACBASE décale automatiquement la deuxième pour insérer un attribut. La grille générée est donc déphasée par rapport à la description dans Dialogue la DLL produite localement. Là encore, ce problème peut être résolu en mettant en phase la description de l'écran et la grille produite par PACBASE, c'est à dire en décalant la deuxième zone d'un caractère.

  Ceci n'entraîne pas de modifications au niveau de l'application, puisque les grilles produites par PACBASE sont inchangées. Il est également possible d'introduire la ligne 021 dans le fichier ENVIRON.PRM pour que dans un tel cas, la mise à jour de la description de l'écran s'effectue de manière automatique.

- L'application à habiller modifie dynamiquement les attributs des zones. Dans ce cas, vérifier que le mode d'apprentissage utilisé est adapté à cette application. Pour cela, consulter le chapitre "Installation", sous-chapitre "Procédure d'installation", section "Boîte Paramètres d'installation PAW", abordant le thème de la sélection d'un mode d'apprentissage.

Si tous ces points sont vérifiés et, le cas échéant, corrigés, vous devez utiliser les outils de diagnostic de PAW :

- lancer PAW sur un poste de développeur (D) sur la ligne 024 du fichier ENVIRON.PRM et accéder à l'écran incriminé.
- ouvrir le menu **Développeur**. L'activation du choix ***Reconnaissance*** déclenche l'ouverture d'une fenêtre de diagnostic. La reconnaissance comporte trois phases. Vérifier si la DLL correspondant à l'écran est contrôlée dans la phase 3. Si oui, le message indique la cause de non reconnaissance qui provient d'un écart (caractère d'un libellé différent, décalage d'un champ) entre le contenu de l'écran reçu (Message) et la description locale de cet écran (DLL). Dans le cas d'une différence de caractères ou d'attributs, l'examen d'une trace de GSCOM permet de savoir si la transcodification des caractères effectuée lors des communications est en cause ou non.

Sinon, refaire le test en isolant la DLL associée à l'écran dans un répertoire afin de réduire le travail de reconnaissance à la phase 3. Attention : il est possible qu'un écran dont la DLL a ainsi été isolée soit reconnu alors qu'il ne l'était pas en fonctionnement normal. Dans ce cas, contacter l'équipe support technique.

Le choix ***Reconnaissance en fichier*** déclenche l'écriture du diagnostic dans le fichier PAW.LOG situé dans le répertoire de personnalisation.

Le choix ***Arbre de décision*** déclenche l'écriture de l'arbre de décision dans le répertoire des exécutables. Ces deux derniers fichiers, ainsi que la trace de GSCOM sont à fournir à l'équipe de support technique en cas de problème persistant.

- Confusion entre deux écrans
  Le système peut avoir des difficultés pour discriminer des DLL dont les découpages en zones protégées et saisissables sont superposables, c'est-à-dire dont les zones similaires ont mêmes positions et mêmes longueurs.

  PAW ne peut pas faire la distinction entre les libellés fixes et les zones protégées. Ainsi, si un libellé fixe et une zone protégée appartenant à deux écrans distincts possèdent un contenu identique, il existe un risque de confusion entre ces deux écrans. Pour éliminer ce problème, il est nécessaire de modifier en longueur ou en contenu le libellé fixe ou la zone protégée.

- Problème de chargement d'une DLL
  En cas d'impossibilité de chargement d'une DLL, PAW envoie un message indiquant le type d'erreur :
  - La DLL est introuvable; elle n'existe pas dans le répertoire.
  - Erreur système avec code système (se reporter à la documentation technique système).

- La DLL n'est pas du type escompté. PAW a tenté de charger une DLL présente dans le répertoire CHEMINS-ECRANS et la DLL n'est pas conforme à ce que vous attendez. Ceci provient soit de la présence de DLL d'un autre type (listes de valeurs externes, etc), soit de l'homonymie d'une DLL d'écran avec une DLL du système. Vérifier alors les homonymes. Une des DLL utilisées par PAW peut avoir le même nom qu'une DLL système (par exemple sous WINDOWS: USER.DLL).

5. Fonctions avancées

5.1. Listes de valeurs externes

Les listes de valeurs externes permettent d'alimenter le choix ***Aide sur les valeurs*** du menu ***Aide*** d'une rubrique ou le menu ***Enchaînement*** d'une rubrique déclarée code opération (voir section "Mise en correspondance et enrichissement des caractéristiques des rubriques") avec des valeurs ne provenant pas des descriptions de rubriques (-D) au niveau du dialogue sur serveur (valeurs internes), comme c'est le cas par défaut.

Si le dialogue utilise des valeurs externes au lieu des informations contenues dans les descriptions de rubriques, vous pouvez donc générer, grâce à PAWGEN et PAWLIS, une version des fichiers d'habillage prenant en compte ces valeurs externes.

Sur le même principe, les listes de valeurs externes vous permettent aussi d'associer à une touche fonction des valeurs qui ne sont pas directement décrites dans le dialogue. PAWGEN génère en effet une zone "touches de fonctions" de code PFKEY_ (où _ désigne un blanc) dans la description locale de chaque écran. Ces valeurs externes s'afficheront dans le menu ***Enchaînement***.

La prise en compte des valeurs externes repose sur deux opérations qui peuvent être effectuées dans n'importe quel ordre. Les deux opérations doivent avoir été menées à leur terme avant tout test ou toute mise en exploitation :

- première opération :
  - obtention en local d'un fichier source formaté contenant les valeurs externes. La récupération du fichier de valeurs ou sa création, ainsi que son formatage sont à la charge du développeur,
  - génération en local des listes de valeurs externes par PAWLIS,
- seconde opération :
  - création d'un fichier de paramétrage pour mettre en correspondance les rubriques ou les touches fonction avec leur propre liste de valeurs externes et indiquer, pour les rubriques, si ce sont des codes opération ou si elles pourront devenir protégées par modification de leurs attributs,
  - génération des fichiers d'habillage par PAWGEN.

5.1.1. Structure des fichiers sources de valeurs externes

Les enregistrements du fichier source local contenant les valeurs externes, que nous désignerons par LISTVAL.DAT, doivent respecter la structure suivante :

[NOM-LISTE**][**LONGUEUR-VALEUR**][**VALEUR**][**LIBELLE-VALEUR**]**

où

- NOM-LISTE indique le nom de la liste de valeurs externes, (fichier qui prendra automatiquement l'extension .DLL) où sera stocké l'enregistrement. Ce nom doit posséder 8 caractères de longueur (compléter par des "blancs" s'il possède moins de 8 caractères). Le même NOM-LISTE sera placé en en-tête de tous les enregistrements devant être associés à une même rubrique ou touche de fonction. Les enregistrements possédant en en-tête le même NOM-LISTE doivent être consécutifs dans le fichier source de valeurs externes.

- LONGUEUR-VALEUR permet d'indiquer sur deux caractères, la longueur, en nombre de caractères, de la valeur définie dans l'enregistrement. Cette longueur peut prendre les valeurs entières comprises entre 01et 99.

- VALEUR est la valeur qui sera affichée dans la boîte d'***Aide sur les valeurs*** ou le menu ***Enchaînement***. La longueur de cette valeur doit être en accord avec le paramètre LONGUEUR-VALEUR décrit précédemment.

- LIBELLE-VALEUR est le libellé associé à la valeur externe. Ce libellé apparaîtra également dans la boîte d'***Aide sur les valeurs*** ou le menu ***Enchaînement***. La longueur du libellé est libre, dans la mesure où l'enregistrement total ne doit pas dépasser 255 caractères.

Remarque Il est possible de créer de multiples fichiers sources de valeurs externes, à partir du moment où leur structure est en accord avec les principes énoncés dans cette section.

Exemple d'un fichier qui fournira les valeurs externes affichées dans la boîte ***Aide sur les valeurs***, concernant les régions et les pays :

RUBREGIO0201AIN

RUBREGIO0240LANDES

RUBPAYS 01FFRANCE

RUBPAYS 02GBANGLETERRE

Exemple d'un fichier qui fournira, dans le menu ***Enchaînement***, les valeurs externes associées à la touche fonction F1 :

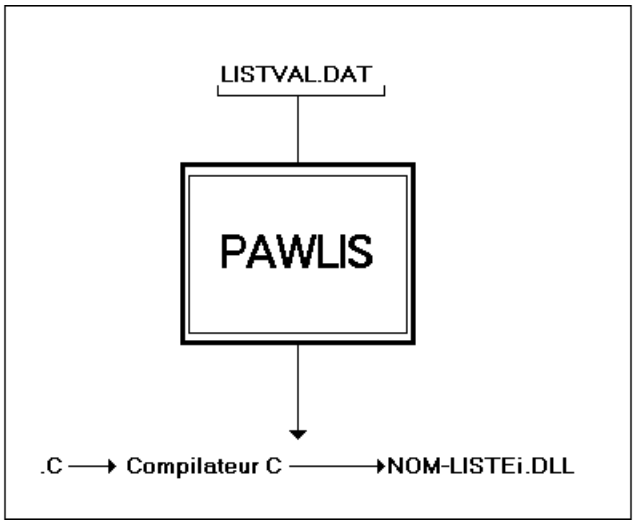
PFKEY1 0201Même écran

PFKEY1 0200Fin de conversation

PFKEY1 02A1Test

PFKEY1 0202Suite

5.1.2. Génération en local des listes de valeurs externes



A partir du fichier LISTVAL.DAT, le développeur va générer grâce à PAWLIS**,** une série de listes de valeurs externes, qui seront automatiquement placées dans le sous-répertoire \LISTE, lié au répertoire spécifié lors de l'installation par le paramètre CHEMIN-BASE. Les fichiers issus de la génération ont les extensions .C et .DEF; ils serviront de fichiers en entrée pour la phase de compilation.

Pour générer les listes de valeurs externes, le développeur doit utiliser PAWLIS, en indiquant comme paramètre le nom du fichier LISTVAL.DAT, c'est-à-dire :

PAWLIS LISTVAL.DAT

Les fichiers d'exemple étudiés précédemment fourniront, une fois traités par PAWLIS et compilés, les fichiers RUBREGIO.DLL, RUBPAYS.DLL et PFKEY1.DLL.

5.1.3. Compilation et vérification des fichiers obtenus

La compilation des fichiers obtenus est déclenchée par une procédure batch générée par PAWLIS.EXE.

On lance les compilations par la commande PAWPROC.

La compilation des fichiers .C a lieu, suivie de l'édition des liens pour produire des .DLL. Ceux-ci devront être copiés par le développeur dans le répertoire correspondant au paramètre CHEMIN-LIST de la procédure d'installation.

Le compte-rendu des compilations s'affiche à l'écran. Le développeur peut le router sur un fichier texte consultable a posteriori, et vérifier qu'il ne contient pas de message d'erreur (chercher la chaîne "error"). En cas de présence de tels messages, consulter le chapitre "Gestion des erreurs".

5.1.4. Mise en correspondance et enrichissement des caractéristiques d'une rubrique

Cette mise en correspondance et cet éventuel enrichissement des caractéristiques d'une rubrique sont à la charge du développeur et s'effectuent par l'intermédiaire d'un fichier que nous appellerons ici RUB-LIST.DAT. Ce fichier permet de mettre en correspondance la rubrique ou la touche fonction avec sa liste de valeurs externes et d'indiquer, le cas échéant, si la rubrique est un code opération ou si elle peut devenir protégée alors qu'elle est initialement saisissable.

Ce fichier sera ultérieurement utilisé par PAWGEN pour générer des .DLL d'écrans tenant compte des listes de valeurs externes. Le fichier RUB-LIST.DAT est composé d'une série d'enregistrements, qui doivent posséder la structure suivante :

[BIB**][**SESS**][**DIALOG**][**ECR**][**RUB**][**NOM-LISTE**][**VAL1**][**VAL2**]**

où

- BIB indique le code de la bibliothèque pour les écrans de laquelle la liste de valeurs externes sera utilisée. Ce code doit être constitué de 3 caractères.
- SESS indique le code de la session pour les écrans de laquelle la liste de valeurs externes sera utilisée. Ce code doit être constitué de 5 caractères.
- DIALOG indique le code du dialogue pour les écrans duquel la liste de valeurs externes sera utilisée. Ce code doit être constitué de 2 caractères.
- ECR indique le code de l'écran pour lequel la liste de valeurs externes sera utilisée. Ce code doit être constitué de 4 caractères.
- RUB indique le code de la rubrique pour laquelle la liste de valeurs externes sera utilisée. Ce code doit être constitué de 6 caractères.
- NOM-LISTE indique le nom de la liste (fichier d'extension .DLL**)** contenant les valeurs externes. Ce code ne doit pas dépasser 8 caractères de longueur.
- VAL1 **(**OPER ou blanc sur 4 caractères**) :**
  - OPER indique que la rubrique sert de code opération. Les valeurs associées remontent dans le menu ***Enchaînement*** (voir le Manuel de l'utilisateur PAW). Un message signale le cas échéant la présence de plus d'un code opération.
  - blanc indique que la rubrique ne sert pas de code opération.

- VAL2 (VARPRO ou blanc sur 6 caractères) :
  - VARPRO indique que la rubrique est initialement saisissable, mais qu'elle peut être protégée par la suite, par modification dynamique des attributs. VARPRO est à utiliser en corrélation avec le mode d'apprentissage des écrans sélectionné. Pour obtenir de plus amples détails à ce sujet, consulter le chapitre "Installation", sous-chapitre "Boîte Paramètres d'installation PAW", section "Cadre Mode d'apprentissage".
  - blanc exclut la modification dynamique des attributs.

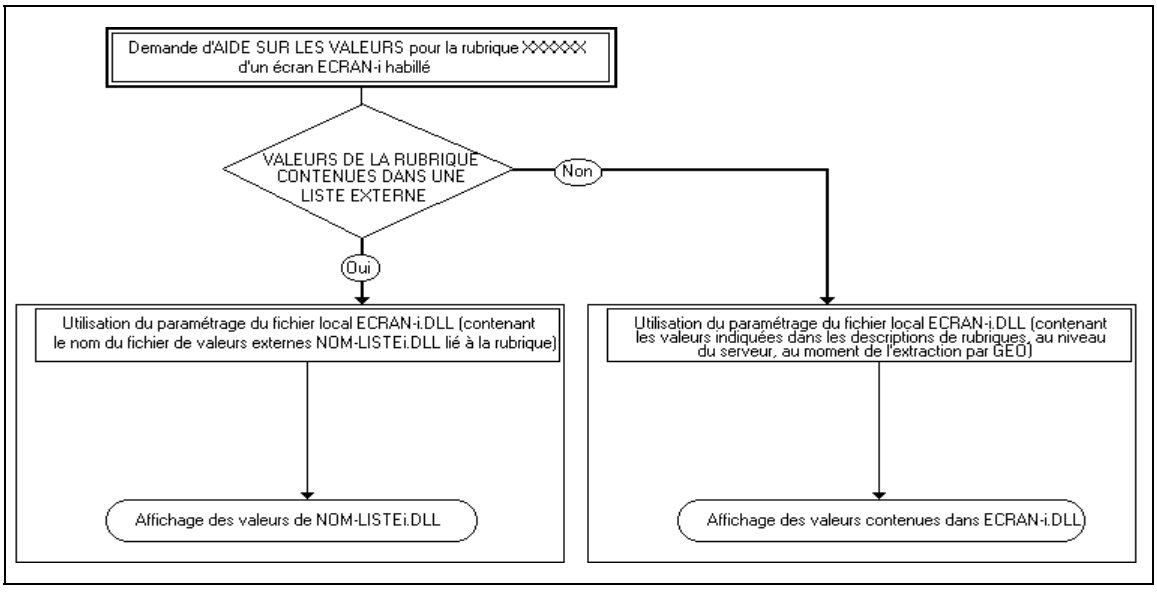
Remarque Pour chacun des éléments décrits ci-dessus, compléter par des blancs pour atteindre la longueur indiquée.

Il est possible de remplacer CODE-BIB, CODE-SESS, CODE-DIAL et CODE-ECR par respectivement *******, *********, ****** et ********. L'utilisation de ce caractère générique permet respectivement d'associer la liste de valeurs externes à toutes les bibliothèques, toutes les sessions, tous les dialogues ou tous les écrans.

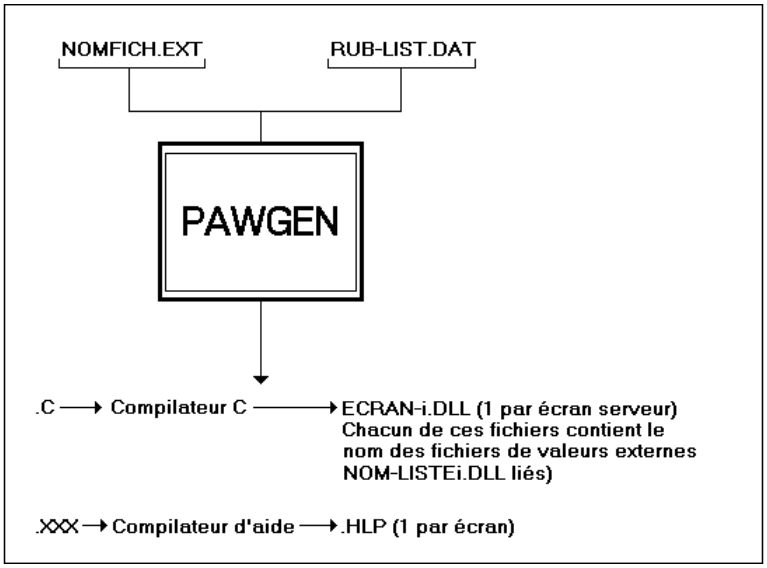
Par exemple, l'enregistrement représenté ci-dessous signifie que les valeurs de la liste RUBREGIO alimenteront la rubrique REGION dans l'ensemble des écrans du dialogue D1 de n'importe quelle session de la bibliothèque B01. Dans tous les autres cas, ce sont les valeurs décrites au niveau du dialogue qui seront automatiquement utilisées, dans le cas où ces valeurs existent (voir graphe ci-après).

B01*****D1****REGIONRUBREGIO

Le schéma ci-dessous illustre dans quel cas les valeurs externes seront prises en compte, suite à une demande d'aide sur les valeurs d'une rubrique.



5.1.5. Génération des fichiers d'habillage par PAWGEN



Le fichier RUB-LIST.DAT étant réalisé, il est possible de procéder à l'habillage en suivant la procédure décrite dans le chapitre "Habillage d'une application". La seule différence réside en la commande de lancement de PAWGEN. Au lieu de taper PAWGEN NOMFICH.EXT comme il est décrit au sous-chapitre "Génération des sources des écrans à habiller" du chapitre "Habillage d'une application", il faut utiliser la commande :

PAWGEN NOMFICH.EXT RUB-LIST.DAT

L'ajout de RUB-LIST.DAT à la commande de lancement de PAWGEN influe sur les .DLL d'écrans produits. En effet, au lieu d'écrire les valeurs des rubriques dans ceux-ci, PAWGEN y inscrit des références à des listes de valeurs externes, en se basant sur les informations contenues dans RUB-LIST.DAT.

Du fait que les fichiers générés par PAWGEN ne contiennent que le nom des listes de valeurs externes, il est possible de mettre à jour ces dernières sans avoir à relancer PAWGEN. Il est seulement nécessaire de les régénérer par PAWLIS après avoir modifié leur contenu.

5.2. Personnalisation de l'aide interactive sur PAW

Par défaut, l'aide fournie à l'activation du choix ***Aide sur les touches*** du menu ***Aide*** est la réplique exacte du contenu du Manuel de l'utilisateur qui vous est fourni en même temps que PAW. Il est possible de personnaliser cette aide, les mouvements PACBASE qui y correspondent étant également fournis.

Les mouvements liés au choix ***Aide sur les touches*** sont automatiquement copiés lors de l'installation dans le répertoire lié au paramètre CHEMIN-EXE de l'installation. Ils sont rassemblés dans le fichier 033MVT.PAC pour la version française, dans le fichier 001MVT.PAC pour la version anglaise, et 034MVT.PAC pour la version espagnole.

Les mouvements fournis correspondent :

- à la définition d'un dialogue,
- à la définition, à la liste des rubriques de l'écran (-CE**)** et à la documentation généralisée (-G**)** de celui-ci,
- aux définitions de rubriques, le nom en clair de chaque rubrique étant destiné à mettre à jour l'index de l'aide,
- à la définition et à la description d'un texte contenant les informations du Manuel de l'utilisateur PAW, découpé en divisions. Chacune des divisions est liée à une rubrique de l'écran au niveau de la documentation généralisée de ce dernier.

Par défaut, l'ensemble des mouvements sont préfixés par "PW". Il est possible de changer ce préfixe par l'intermédiaire d'une fonction de recherche/remplacement globale. Il est également possible de modifier le contenu de l'aide en modifiant directement les mouvements. C'est cependant la procédure de modification après intégration des mouvements à la base PACBASE, plus conviviale, qui sera développée ci-après.

La remontée sur le serveur des mouvements, ainsi que leur intégration à la base PACBASE par l'intermédiaire de la procédure UPDT sont à la charge du développeur. Il est à noter que les mouvements peuvent être intégrés à n'importe quelle base et à n'importe quelle bibliothèque, au choix du développeur.

Remarque La carte "étoile" n'est pas fournie. Ne pas omettre de la créer en tête du fichier de mouvements avant de lancer UPDT. En cas d'oubli, la procédure de mise à jour échouerait.

5.2.1. Modifier le texte d'un paragraphe existant

A partir du nom en clair des rubriques composant l'écran et de la documentation généralisée de ce dernier, choisir le texte à modifier. Mettre à jour ce texte (modifications, traduction, suppression et ajouts d'informations).

5.2.2. Ajouter une ou plusieurs divisions

Créer un nouveau texte d'aide signifie :

- mettre à jour l'écran issu de l'intégration des mouvements fournis sur la disquette d'installation, en y ajoutant une nouvelle rubrique dont le nom en clair sera utilisé comme entrée de l'index de l'aide.
  La rubrique ajoutée doit nécessairement se trouver après les rubriques standard, dans une séquence ASCII. Par ailleurs, la plage de PW0010 à PW0900 est réservée pour d'éventuelles modifications de PAW. Pour éviter tout risque d'erreur, il est donc vivement conseillé d'employer pour toute nouvelle rubrique un préfixe différent du préfixe standard (PW par défaut), et venant après celui-ci dans un classement ASCII.

- créer une nouvelle division de texte et y saisir les informations à ajouter à l'aide sur les touches,

- chaîner le nouveau texte à la rubrique par l'intermédiaire de la documentation généralisée de l'écran.

5.2.3. Regénération de l'aide interactive

Une fois le nouveau texte au point (texte créé ou seulement modifié), faire subir à l'écran issu de l'intégration des mouvements d'aide la procédure d'habillage décrite au chapitre "Habillage d'une application". Procéder à l'extraction sur site central, au transfert du fichier d'extraction sur micro-ordinateur. L'utilisation de PAWGEN pour la génération de l'aide interactive est légèrement différente de la description qui en est faite au sous-chapitre "Génération des sources des écrans à habiller" du chapitre "Habillage d'une application". Il faut taper la commande suivante :

PAWGEN NOMFICH.EXT *

Le caractère "*" indique que l'on génère l'aide sur les touches de l'application. PAWGEN ne fournit alors pas de source C**,** mais uniquement un fichier d'extension .IPF ou .RTF qui devra être compilé avec le compilateur d'aide. Le résultat de la compilation sera un fichier d'extension .HLP. Consulter la section "Compilation et vérification des fichiers obtenus" du sous-chapitre "Listes de valeurs externes" pour de plus amples renseignements à ce sujet.

Pour tester la nouvelle version de paramétrage produite, le développeur copie l'ensemble des fichiers sur son répertoire de test (CHEMIN-ECRANS) par la commande COPY CHEMIN-BASE\SESSION\BIB\PAW_HELP.HLP CHEMIN-ECRANS. Il sera nécessaire de renommer le fichier PAW_HELP.HLP en PW_HLPEN pour la version anglaise, PW_HLPES pour la version espagnole ou PW_HLPFR pour la version française.

Le transfert sur le poste de l'utilisateur s'effectue par une copie du paramétrage produit sur le répertoire lié au CHEMIN-ECRANS de l'installation du poste de l'utilisateur.

5.3. Automatisation des tâches : .BAT

La plupart des phases d'habillage peuvent être automatisées par des fichiers batch, d'extension .BAT sous DOS. Les exemples ci-après ne peuvent être utilisés tels quels pour n'importe quelle configuration : ils doivent être adaptés au poste du développeur (nom des répertoires, des fichiers à traiter...). Précédées de "(COM)", apparaissent des informations ne faisant pas partie du fichier batch, mais qui en permettent une meilleure compréhension.

L'automatisation des tâches suppose une bonne connaissance de la syntaxe des .BAT, qui n'est pas l'objet de ce manuel. Pour obtenir toutes les informations sur ce sujet, consulter la documentation système DOS.

- automatisation de la génération

```
(COM) Cette procédure utilise de 1 à 3 paramètres, à savoir :
(COM)     2 paramètres pour PAWGEN.EXE.
(COM)       1 : Nom du fichier extrait du site central
(COM)       2 : Nom du fichier d'association entre listes et rubriques
(COM)           ou * pour générer l'aide sur les touches.
(COM)     1 paramètre pour PAWLIS.EXE.
(COM)       3 : Nom du fichier des listes externes.

@echo off                          (COM) Suppression de l'echo et
cls                                (COM) remise à blanc de l'écran.
echo     +-------------------------------------------------------------------------+
echo     ¦                                                                         ¦
echo     ¦    Procédure PAWG de génération des fichiers                            ¦
echo     ¦    d'aide et des écrans logiques (DLL)                                  ¦
echo     ¦                                                                         ¦
echo     ¦    3 paramètres peuvent être utilisés.                                  ¦
echo     ¦        1 : Nom du fichier extrait de la base centrale.                  ¦
echo     ¦            OBLIGATOIRE.                                                 ¦
echo     ¦        2 : Soit le nom du fichier d'association entre rubriques         ¦
echo     ¦            et listes de valeurs externes.                               ¦
echo     ¦            Soit le caractère * qui indique qu'on veut générer           ¦
echo     ¦            l'aide sur les touches.                                      ¦
echo     ¦            FACULTATIF.                                                  ¦
echo     ¦        3 : Si on génére des listes de valeurs, nom du fichier           ¦
echo     ¦            contenant les valeurs.                                       ¦
echo     ¦            FACULTATIF.                                                  ¦
echo     ¦                                                                         ¦
echo     +-------------------------------------------------------------------------+

:PARAM                             (COM)   Test des paramètres :
if p%1     == p goto FIN           (COM)       Pas de paramètre :FIN

:PGEN                              (COM)   Exécution de PAWGEN.
echo .
cd REPEXE                          (COM)   Le répertoire courant est celui des EXE
if NOT p%2 == p goto PGEN_2

PGEN_1
echo  *** Exécution de PAWGEN AVEC UN PARAMETRE
PAWGEN  %1     >  PAWGEN.lis       (COM) Le compte rendu est écrit dans PAWGEN.LIS
if errorlevel 1 goto STOP_GEN      (COM) Si le code retour de PAWGEN.EXE est
                                   (COM) différent de zéro, la procédure est
                                   (COM) interrompue.
goto COMPIL                        (COM) Sinon, on enchaîne sur la compilation.

PGEN_2
echo  *** Exécution de PAWGEN AVEC DEUX PARAMETRES
PAWGEN  %1 %2  >  PAWGEN.lis       (COM) Le compte rendu est écrit dans PAWGEN.LIS
if errorlevel 1 goto STOP_GEN

:COMPIL
echo .
echo  *** Construction des .DLL et des .HLP ..
echo ..
call PAWPROC

echo .
echo *** Traitement des listes  ***
if p%3% == p goto CPT_RENDU        (COM) Pas de listes à traiter
cd REPEXE                          (COM)   Le répertoire courant a pu changer.
PAWLIS %3
if errorlevel 1 goto STOP_LIS      (COM) En cas d'erreur, on arrête le traitement

echo  *** Construction des .DLL de listes externes ..
call PAWPROC
goto CPT_RENDU

:STOP_GEN
echo    Erreur au cours de l'exécution de PAWGEN
```

```
echo    Arrêt des traitements
goto CPT_RENDU

:STOP_LIS
echo    Erreur au cours de l'exécution de PAWLIS
echo    Arrêt des traitements
goto CPT_RENDU

:CPT_RENDU
echo Edition du compte-rendu.
EDIT %1.lis             (COM)   Remplacer EDIT par le nom de l'éditeur
goto FIN

:FIN
```

- automatisation de la mise en exploitation

```
)
(COM) Cette procédure utilise 2 paramètres, à savoir :
(COM)     Le code de la bibliothèque et le numéro de session (9999H
(COM)     pour la session courante).

@echo off                           (COM) Suppression de l'echo et
cls                                 (COM) remise à blanc de l'écran.

rem Pas de paramètre -> pas de mise en exploitation...
if p%1 == p goto KO
if p%2 == p goto KO
goto OK

:KO
echo ...
echo Passer 2 paramètres SVP : code BIBLIOTHEQUE et SESSION !
echo ...
pause
goto FIN

:OK
echo .
echo ---» Mise en exploitation de %2\BIB_%1...
echo .
c:
cd PAWBAS\%2\bib_%1         (COM) On se place sur le répertoire contenant les fichiers.
copy *.dll PAWTST           (COM) Copie des DLL et des HLP sur le répertoire de test
copy *.hlp PAWTST           (COM) du poste du développeur ou sur le répertoire
                            (COM) du serveur contenant les fichiers d'exploitation.

echo .
echo *** Mise en exploitation terminée !!! ***
echo .

pause
goto FIN

:FIN
```

- automatisation de la génération de listes de valeurs externes

```
@echo off                           (COM) Suppression de l'echo et
cls                                 (COM) remise à blanc de l'écran.
echo     +-------------------------------------------------------------------------+
echo     ¦                                                                         ¦
echo     ¦    Procédure PAWL de génération des listes de valeurs externes          ¦
echo     ¦                                                                         ¦
echo     ¦    Paramètre :                                                          ¦
echo     ¦            Nom du fichier contenant les valeurs                         ¦
echo     ¦            OBLIGATOIRE.                                                 ¦
echo     ¦                                                                         ¦
echo     +-------------------------------------------------------------------------+

:PARAM                              (COM)   Test des paramètres :
if p%1     == p goto FIN            (COM)       Pas de paramètre :FIN

echo *** Traitement des listes  ***
cd REPEXE                           (COM)   Le répertoire courant est celui des EXE
PAWLIS %3
if errorlevel 1 goto STOP_LIS       (COM) En cas d'erreur, on arrête le traitement
echo  *** Construction des .DLL de listes externes ..
call PAWPROC >> PAWGEN.lis
goto CPT_RENDU

:STOP_LIS
echo    Erreur au cours de l'exécution de PAWLIS
echo    Arrêt des traitements
goto CPT_RENDU

:CPT_RENDU
echo Edition du compte-rendu.
EDIT %1.lis             (COM)   Remplacer EDIT par le nom de l'éditeur
goto FIN

:FIN
```

5.4. Configuration du clavier

5.4.1. Généralités

Lors de la première utilisation de PAW, un fichier de configuration du clavier est automatiquement créé. Ce fichier se trouve dans le répertoire de personnalisation et se nomme PAW_KBRD.PRM. Ce fichier détermine notamment quelle touche sera affectée à la transmission des données. Ce fichier est éditable et les lignes qui suivent indiquent comment changer la touche de transmission, ou en ajouter une. En cas d'erreur, il est ensuite toujours possible de détruire le fichier PAW_KBRD.PRM, PAW le recréant dans sa forme initiale à la session suivante.

Le fichier PAW_KBRD.PRM contient la description des correspondances entre une touche sur le serveur et une combinaison de touches du PC.

Chaque correspondance est codée sous forme d'une ligne comportant :

- le codage sur 18 caractères. Le premier caractère indique si la combinaison est active (S) ou inactive (U). Ces 18 caractères sont les seuls significatifs.
- des champs documentaires :
  - Serveur : action serveur (ex : transmission = ENT),
  - PC : touche PC éventuellement accompagnée des touches Maj, Ctrl et Alt.

La touche de transmission par défaut est la touche Entrée.

Cette assignation est matérialisée par la présence de la ligne suivante dans le fichier PAW_KBRD.PRM :

codage	Serveur	PC	Maj	Ctrl	Alt
S002580D0000000191	**ENT**	NEWLINE			

Pour transmettre avec la touche Ctrl, activer la ligne suivante en remplaçant le U par un S:

codage	Serveur	PC	Maj	Ctrl	Alt
U00258110000000191	**ENT**	CONTROL			

Codage	**Serveur**	**PC**	**Maj**	**Ctrl**	**Alt**
S00258700000105081		**F01**		**1**	
S00258720000105051		**F03**		**1**	
S00258740000105061		**F05**		**1**	
S002588C0000000191	**ENT**	**ENTER**			
S002580D0000000191	**ENT**	**NEWLINE**			
S002582C0000000171	**PA1**	**PRINTSCRN**			
S00258910000000181	**PA2**	**SCRLLOCK**			
S00258130000000131	**CLS**	**PAUSE**			
S00258001010100491	**FLC**	**e (0x65)**		**1**	
S00258700000000201	**PF01**	**F01**			
S00258710000000211	**PF02**	**F02**			
S00258720000000221	**PF03**	**F03**			
S00258730000000231	**PF04**	**F04**			
S00258740000000241	**PF05**	**F05**			
S00258750000000251	**PF06**	**F06**			
S00258760000000261	**PF07**	**F07**			
S00258770000000271	**PF08**	**F08**			
S00258780000000281	**PF09**	**F09**			
S00258790000000291	**PF10**	**F10**			
S002587A0000000301	**PF11**	**F11**			
S002587B0000000311	**PF12**	**F12**			
S002587C0000000321	**PF13**	**F13**			
S00258700001000321	**PF13**	**F01**	**1**		

```
S002587D0000000331  PF14  F14
S00258710001000331  PF14  F02  1
S002587E0000000341  PF15  F15
S00258720001000341  PF15  F03  1
S002587F0000000351  PF16  F16
S00258730001000351  PF16  F04  1
S002587F0000000361  PF17  F16
S00258740001000361  PF17  F05  1
S002587F0000000371  PF18  F16
S00258750001000371  PF18  F06  1
S002587F0000000381  PF19  F16
S00258760001000381  PF19  F07  1
S002587F0000000391  PF20  F16
S00258770001000391  PF20  F08  1
S002587F0000000401  PF21  F16
S00258780001000401  PF21  F09  1
S002587F0000000411  PF22  F16
S00258790001000411  PF22  F10  1
S002587F0000000421  PF23  F16
S002587A0001000421  PF23  F11  1
S002587F0000000431  PF24  F16
S002587B0001000431  PF24  F12  1
S002581B0000000471  REST  ESCAPE
S00258240000000012  HAUT  HOME
S00258230000000022  BAS   END
S00258090001000032  GCHE  TAB  1
S00258090000000042  DRTE  TAB
S00258240000100052  TAV   HOME        1
S00258230000100062  TAR   END         1
S00258080000100072  HOME  BACKSPACE   1
S002582E0000100082  EEOF  DELETE      1
S00258080000000092  ZLIGS BACKSPACE
S002580D0000000102  ATTR  NEWLINE
S00258250000000112  SYST  LEFT
S00258270000000122  ATTN  RIGHT
S00258250001000132  CLS   LEFT  1
S00258270001000142  ERA   RIGHT 1
S00258900000000002  NUMLOCK
S002586F0000000002  DIVIDE
S002586A0000000002  MULTIPLY
S002586D0000000002  SUBTRACT
S002586B0000000002  ADD
S002586E0000000002  DECIMAL
S00258600000000002  NUM00
S00258610000000002  NUM01
S00258620000000002  NUM02
S00258630000000002  NUM03
S00258640000000002  NUM04
S00258650000000002  NUM05
S00258660000000002  NUM06
S00258670000000002  NUM07
S00258680000000002  NUM08
S00258690000000002  NUM09
U00258110000000191  ENT   CONTROL
U00258220000002551  PAGEDOWN
U00258210000002551  PAGEUP
U00258240000002551  HOME
U00258230000002551  END
U00258260000002551  UP
```

col.	long.	signification
1	1	: S si touche active : U si touche inactive
2	5	: valeur du message
7	2	: virtual key (hexa)
9	3	: code touche (numérique)
12	3	: booléens pour Ctrl - Alt - Shift
15	3	: identification de la fonction ex: pour fonctions internes, valeurs de 000 à 020
18	1	: 1 si transmission au Host : 2 si fonction interne

CTRL + F1 -> CTRL + F12	F1 … F12
CTRL + MAJ + F1 -> CTRL + MAJ + F12	F13 … F24

5.4.2. Les fonctions locales

Les fonctions locales sont des fonctions préprogrammées, activables par certaines touches du clavier. L'utilisateur définit ces touches dans le fichier PAW_KBRD.PRM en leur associant le numéro de la fonction interne désirée (colonnes 15 à 17).

Ces fonctions sont les suivantes :

n°	fonctions
0	ne fait rien
1	début de champ
2	fin de champ
3	début du champ précédent
4	début du champ suivant
5	début du premier champ de l'écran
6	début du dernier champ de l'écran
7	effacement du début du champ
8	effacement de la fin du champ
9	supprime le caractère précédent
10	début du premier champ de la ligne suivante
11	déplacement d'un caractère vers la gauche
12	déplacement d'un caractère vers la droite
13	désélectionne le caractère précédent
14	désélectionne le caractère suivant
15	impression de l'écran
16	déplacement d'un caractère vers la gauche avec tabulation automatique sur le champ précédent
17	déplacement d'un caractère vers la droite avec tabulation automatique sur le champ suivant
18	rappel de la dernière fonction locale utilisée
19	début du dernier champ de la ligne précédente
20	début du premier champ de la ligne suivante

(Les champs ci-dessus mentionnés doivent être compris comme étant des champs saisissables).

5.4.3. Valeurs des touches (à mettre dans PAW_KBRD.PRM)

Ces valeurs sont à mettre en colonnes 7-8 (virtual key).

Sont prises en compte :

Valeur hexa	Nom de la touche
1b	Escape
2d	Inser
24	Home
2e	Suppr
23	Fin
21	PageUp
22	PageDown
25	Flèche à gauche
26	Flèche en haut
27	Flèche à droite
28	Flèche en bas
6d	Subtract
6f	Divide
6a	Multiply
6b	Add
6e	Decimal
60	num.0
61	num.1
62	num.2
63	num.3
64	num.4
65	num.5
66	num.6
67	num.7
68	num.8
69	num.9

5.5. Clavier Logiciel

5.5.1. Caractéristiques

Le clavier logiciel est une fenêtre qui permet d'effectuer quatre types d'action à l'aide de la souris : actions de transmission standard (transmission simple, effacement de l'écran, restauration, demande d'impression), et actions présentes dans les menus ***Action***, ***Enchaînement*** et ***Scripts***.

Chaque type d'action correspond à un type de bouton (bouton standard, d'action, d'enchaînement et de scripts) dont l'utilisateur paramètre l'affichage dans le menu ***Options***, choix ***Préférences***.

Les boutons sont par défaut des boutons-poussoirs. Il est cependant possible de les remplacer par des boutons porteurs d'icône. Les icônes sont stockées dans un fichier de type DLL qui se trouve dans le répertoire des fichiers paramètres de l'utilisateur (ligne 028 du fichier d'environnement) et dont le nom (radical sans extension) est indiqué à la ligne 029 du fichier d'environnement.

Lors de l'installation de PAW sur un poste de développeur, le programme copie sur le répertoire de base de l'utilisateur les fichiers contenant les icônes des boutons de transmission standard, ainsi qu'un fichier source C de la DLL et le fichier de procédure exemple (DOICOPAW.BAT) pour la compilation et l'édition des liens de DLLs d'icônes. Ce fichier peut être modifié ou adapté à un autre compilateur que celui utilisé (MICROSOFT C 8.00). Dans l'étape d'édition des liens, il faut toujours utiliser les fichiers LIBENTRY.OBJ, MDLLCEW.LIB (ou LDLLCEW.LIB) et LIBW.LIB.

5.5.2. Modification du jeu d'icônes disponibles

Vous pouvez personnaliser les icônes associées au clavier logiciel. Ces icônes sont stockées dans un fichier DLL (par exemple ICOPAW.DLL dans le répertoire des fichiers de personnalisation) résultant de la compilation et de l'édition des liens des fichiers suivants :

ICOPAW.C: source C

ICOPAW.RC: ressources WINDOWS

Le fichier ICOPAW.C ne doit pas être modifié.

Le fichier ICOPAW.RC contient :

- une ligne INCLUDE <WINDOWS.H>, qui ne doit pas être modifiée.
- une série de lignes de définition pour les boutons standard, qui ne doivent pas être modifiées.
- des lignes d'association entre des icônes (fichiers d'extension ICW) et des nombres, représentés chacun par une chaîne de caractères (par exemple ICO_ENT).

5.5.2.1. Ajout de nouvelles icônes dans le fichier ICOPAW.RC

- Etape 1 : Introduire une nouvelle ligne d'association par icône, suivant la syntaxe :

1000 ICON NEWICON.ICW

où :

- 1000 est le numéro de ressource identifiant l'icône, compris entre 1000 et 9999.
- ICON est un mot-clé.
- NEWICON.ICW est le code du fichier de description de l'icône (fichier à créer au moyen d'outils tels que PAINTBRUSH et SDKPAINT par exemple, et que l'on aura copié dans le répertoire de base du développeur CHEMIN-BASE).

- Etape 2 : Exécuter le fichier de procédure DOICOPAW.BAT pour fabriquer la DLL, en indiquant le nom du fichier DLL en paramètre.

Remarque : Pour pouvoir être utilisé, le fichier DLL doit porter son nom initial, soit celui donné lors de sa création par DOICOPAW.

- Etape 3 : Recopier le fichier DLL produit dans le répertoire des fichiers de personnalisation.

Si l'on ne désire pas écraser l'ancien fichier DLL par le fichier obtenu tout en utilisant le nouveau paramétrage d'icônes, il suffit :

- soit de renommer l'ancien fichier DLL,
- soit de modifier la ligne 029 du fichier ENVIRON.PRM en y inscrivant le nom du nouveau fichier.

Remarque : On ne peut pas attribuer un même nombre (ou une même chaîne de caractères) à deux fichiers ICW différents.

5.5.2.2. Modification de l'icône d'un bouton standard

Chacun des boutons standard est référencé dans le fichier ICOPAW.RC et associé à un nombre. Pour associer une nouvelle icône à ce bouton, vous devez :

1 Copier le fichier icône (d'extension ICW) dans le répertoire de base du développeur CHEMIN-BASE (répertoire de travail),

2 Dans le fichier ICOPAW.RC, associer le fichier icône au nombre représentant le bouton concerné,

3 Lancer le fichier de procédure DOICOPAW.BAT pour fabriquer la DLL, en indiquant le nom du fichier DLL en paramètre (par exemple NEWICOPAW.DLL),

4 Recopier le fichier DLL produit dans le répertoire des fichiers de personnalisation.

5.5.2.3. Association d'une icône à un bouton d'action ou d'enchaînement

Les boutons non standard proviennent soit de la base PACBASE, soit d'une liste de valeurs externes. Pour associer un de ces boutons à une icône, il faut lui donner un identifiant numérique. Cet identifiant est un nombre de quatre chiffres compris entre 1000 et 9999 placé entre crochets en tête du libellé de la valeur prise en compte dans les menus ***Action*** ou ***Enchaînement***. Lancer ensuite la procédure de fabrication de DLL (DOICOPAW.BAT) et recopier la DLL produite dans le répertoire adéquat.

5.5.2.4. Association d'une icône à un script

Pour associer une icône à un script, placer le code numérique de quatre chiffres identifiant l'icône en tête du libellé du script (dans le fichier d'extension PRM correspondant). Lancer ensuite la procédure de fabrication de DLL (DOICOPAW.BAT) et recopier la DLL produite dans le répertoire adéquat.

Si le chargement de la DLL échoue ou si le programme ne peut afficher l'icône, le bouton est affiché sous sa forme poussoir avec le texte provenant du libellé.

6. Exemples d'habillage PAW

6.1. Portage d'une application MICROFOCUS vers une application habillée par PAW

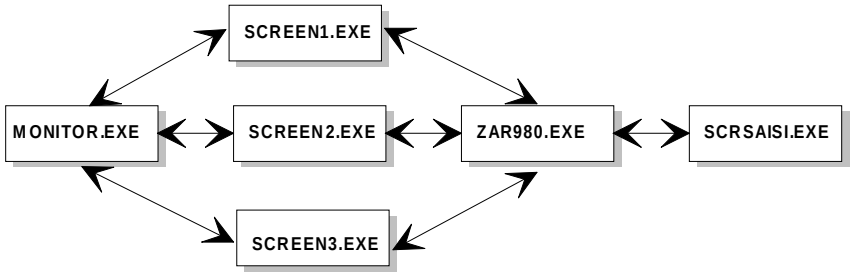
6.1.1. Architecture d'une application Dialogue MICROFOCUS DOS

Une application Dialogue MICROFOCUS DOS standard est constituée des programmes suivants (voir schéma ci-dessous) :

- MONITOR.EXE est le moniteur du dialogue. Il est chargé de gérer l'enchaînement des écrans. C'est le premier programme qui est exécuté lorsqu'une application Dialogue débute. SCREEN2.EXE, SCREEN2.EXE, ... sont les programmes propres aux écrans du dialogue.
- ZAR980.EXE sert d'interface entre les programmes des écrans et les programmes d'entrées/sorties.
- SCRSAISI.EXE est le programme d'entrées/sorties chargé de l'affichage vidéo et de la saisie au clavier.

Les sources Cobol de ZAR980 et SCRSAISI sont fournis par IBM.

Pour générer ces programmes, votre fichier de commande de compilation devrait ressembler à celui donné à la section "Exemples de fichiers de commandes de compilation".



1. Structure d'une application Dialogue MICROFOCUS DOS

6.1.2. Architecture de la même application habillée sous WINDOWS 3

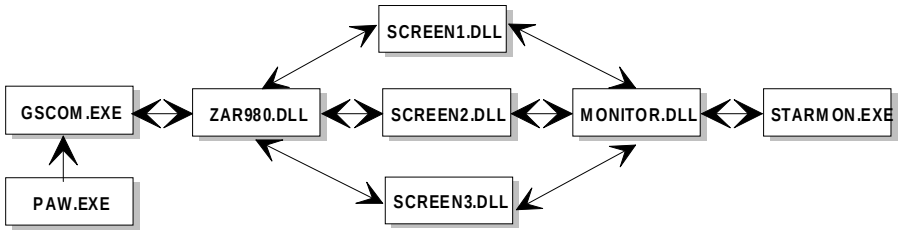
La cinématique (voir schéma ci-après) des appels des programmes d'une application MICROFOCUS habillée sous WINDOWS est identique à celle d'une application MICROFOCUS sous DOS. Cependant le moniteur n'est pas lancé directement, mais par l'intermédiaire de PAW via GSCOM.EXE et STARTMON.EXE.

D'autre part, les programmes ZAR980.EXE et SCRSAISI.EXE sont remplacés par un programme unique ZAR980.DLL. C'est par ce programme, via GSCOM.EXE**,** que se feront les entrées/sorties (affichage et saisie clavier) entre l'application dialogue MICROFOCUS et PAW.

Les sources Cobol ne donneront plus naissance à des .EXE mais à des .DLL (se référer à la documentation du Software Development Kit de WINDOWS pour une description complète des concepts des librairies dynamiques). Ces .DLL ont l'avantage de repousser les limites de taille du code exécutable (sous DOS la taille maximum d'un programme étant d'environ 500 Ko, sous WINDOWS un programme peut être 2 à 3 fois plus important). D'autre part, WINDOWS peut charger simultanément plusieurs programmes de cette taille.

Vous trouverez à la section "Exemples de fichiers de commandes de compilation" un exemple permettant d'obtenir les .DLL pour le moniteur et les programmes des écrans. Cette génération est obtenue à partir des mêmes .OBJ générés lors de la compilation Cobol de l'application sous DOS.

Les programmes PAW, GSCOM, STARTMON et ZAR980 sont fournis par IBM.



2. Structure de la même application habillée sous WINDOWS 3

6.1.3. Remarques et conseils

- L'habillage d'applications MICROFOCUS ne peut se faire que si elles ont été générées à partir de PACBASE.

- S'assurer qu'il n'y a pas d'instruction COBOL "STOP-RUN" dans le source de chaque moniteur. S'il s'en trouve, changer toutes ces instructions par des instructions "GOBACK" à l'aide d'un éditeur de texte. Cette génération est automatique dans les versions récentes de PACBASE.

- Les instructions DISPLAY ne doivent pas être utilisées.

- Afin d'éviter tout conflit au niveau de WINDOWS entre les .DLL des programmes du moniteur et des écrans et les .DLL de l'habillage, prendre soin de renseigner dans la définition de chaque écran un nom de map *différent* de celui du programme.
  Exemple *nom de programme :* ***DO0002***
  nom de ***MAP*** *:* ***DO0002M***

- La DLL ZAR980.DLL utilisée est celle de l'installation. Donc, NE PAS fabriquer de DLL de ce nom à partir du source COBOL servant à produire le ZAR980.EXE de la version plein écran.
  De plus, insérer dans le fichier des paramètres ENVIRON.PRM (dans le répertoire de PAW) une ligne 022 comme suit :

```
--- ENVIRON.PRM ---
001 ...
...
022     Noms externes des DLL        [1]
```

- Le répertoire des .DLL de votre application MICROFOCUS doit être inclus dans les variables d'environnement PATH et COBDIR.

- La relation entre un nom de fichier interne et un nom de fichier externe dans une application MICROFOCUS sous WINDOWS est identique à celle d'une application MICROFOCUS sous DOS, c'est-à-dire basée sur le jeu de variables d'environnement. En conséquence si vous souhaitez exécuter plusieurs instances de PAW sur des moniteurs différents (il est d'ailleurs impossible d'exécuter plus d'une fois PAW sur le même moniteur), prendre garde d'associer aux applications des noms de fichiers internes distincts.
  Exemple :
  *application **APPLI1** nom interne du fichier des libellés d'erreurs **APP1LE***
  *application **APPLI2** nom interne du fichier des libellés d'erreurs **APP2LE**.*

  En revanche, les noms des fichiers externes peuvent être identiques.
  Exemple :
  C:\APPLI1\LE pour le fichier des libellés d'erreurs de l'application 1
  C:\APPLI2\LE pour le fichier des libellés d'erreurs de l'application 2
  avec le réglage des variables d'environnement suivant :
  SET APP1LE=C:\APPL1\LE
  SET APP2LE=C:\APPL2\LE

- Noter que dans les exemples de fichiers de commandes de compilation fournis ci-après, seul le programme du moniteur est linké avec les modules EXTFH.OBJ et EXTERNL.OBJ (modules contenant les routines d'accès COBOL aux fichiers fournis avec le WORKBENCH MICROFOCUS). Ceci permet de réduire la taille de chaque programme d'écran d'une quantité non négligeable d'environ 70Ko.
- Vérifier que le répertoire où se trouve le programme ZAR980.DLL (fourni avec l'environnement PAW) est bien inclus dans la variable PATH de l'AUTOEXEC.BAT.

- Bien que WINDOWS travaille en ANSI, le format des fichiers externes traités par les générateurs PAWLIS et PAWGEN doit être ASCII.

6.1.4. Exemples de fichiers de commandes de compilation

- Exemple de fichier de commande de compilation d'une application dialogue MICROFOCUS sous DOS :

```
         rem --- COMPILATION COBOL ---
cobol ZAR980.CBL ;
cobol SCRSAISI.CBL ;
cobol MONITOR.CBL ;
cobol SCREEN1.CBL ;
cobol SCREEN2.CBL ;
cobol ... autres écrans...

         rem --- EDITION DE LIENS ---
link SCRSAISI.OBJ, SCRSAISI.EXE, NUL,
          /nod LCOBOL.LIB COBAPI.LIB, NUL
link ZAR980.OBJ, ZAR980.EXE, NUL,
          /nod LCOBOL.LIB COBAPI.LIB, NUL
link MONITOR.OBJ EXTFH.OBJ EXTERNL.OBJ XFHNAME.OBJ,
          MONITOR.EXE, NUL, /nod LCOBOL.LIB COBAPI.LIB, nul
link SCREEN1.OBJ , SCREEN1.EXE, NUL,
          /nod LCOBOL.LIB COBAPI.LIB, NUL
link SCREEN2.OBJ , SCREEN2.EXE, NUL,
          /nod LCOBOL.LIB COBAPI.LIB, NUL
link ... autres écrans...
```

- Exemple de fichier de commande de compilation de l'application dialogue MICROFOCUS habillée sous WINDOWS :

```
         rem --- COMPILATION COBOL ---
COBOL MONITOR.CBL ;
COBOL SCREEN1.CBL ;
COBOL SCREEN2.CBL ;
COBOL autres écrans ...

         rem --- EDITION DE LIENS ---
LINK @MONITOR.LNK
LINK @SCREEN1.LNK
LINK @SCREEN2.LNK
LINK autres écrans ...
```

où @**MONITOR.LNK** contient les lignes suivantes :

```
MONITOR+CBLWINL+LIBINIT+EXTFH+EXTERNL+XFHNAME, MONITOR.DLL,,
LCOBOLW+LCOBOL+COBW, MONITOR.DEF /nod /noe;
```

où **MONITOR.DEF** contient les lignes suivantes :

```
LIBRARY         MONITOR
DESCRIPTION     'Monitor PAW and WINDOWS'
EXETYPE         WINDOWS 3.0
CODE            PRELOAD MOVABLE DISCARDABLE
DATA            PRELOAD SINGLE NONSHARED
HEAPSIZE        0
EXPORTS         MONITOR @1
```

où @**SCREEN1.LNK** contient les lignes suivantes :

```
SCREEN1+CBLWINL+LIBINIT,SCREEN1.DLL,,
LCOBOLW+LCOBOL+COBW,SCREEN1.DEF /nod /noe;
```

où **SCREEN1.DEF** contient les lignes suivantes :

```
LIBRARY         SCREEN1
DESCRITPION     'Screen PAW and WINDOWS'
EXETYPE         WINDOWS 3.0
CODE            PRELOAD MOVEABLE DISCARDABLE
DATA            PRELOAD SINGLE NONSHARED
HEAPSIZE        0
EXPORTS         SCREEN1 @1
```

Poursuivre de la même manière pour tous les autres écrans du dialogue.

Remarque Les commandes de compilations et les bibliothèques à utiliser lors de l'édition des liens peuvent varier d'une version du COBOL à une autre. Il faut donc se reporter à la documentation MICROFOCUS. En revanche, Ies fichiers .DEF ne dépendent que de la version de WINDOWS .

6.2. Habillage d'un produit IBM : DSMS

6.2.1. Habillage DSMS : présentation

Un habillage finalisé des écrans DSMS est disponible afin d'offrir aux utilisateurs de ce produit les avantages d'un environnement graphique micro.

Le rôle du développeur peut être réduit à la simple installation de PAW et des fichiers d'habillage et d'aide fournis sur les postes des utilisateurs.

Dans le cas de l'habillage de DSMS, il peut également réaliser des listes de valeurs externes qui permettent d'alimenter le choix ***Aide sur les valeurs*** du menu ***Aide*** ou le menu ***Enchaînement*** (voir le Manuel de l'utilisateur PAW) avec les valeurs préalablement saisies dans les tables DSMS. Ce travail suppose une installation de l'habillage DSMS sur le poste du développeur. Les listes de valeurs réalisées devront être installées par la suite sur les postes des utilisateurs, en plus des fichiers d'habillage et d'aide fournis. Pour des détails sur les listes de valeurs externes, se référer au chapitre "Fonctions avancées", sous-chapitre "Liste de valeurs externes".

L'installation de DSMS habillé comprend celle du **traitement de texte local**. Les *fichiers .DLL* faisant référence aux tables des Langues, des Produits et des Filiales sont automatiquement copiées dans le répertoire des listes de valeurs externes (voir ci-dessous). Si, après l'installation, vous modifiez les chemins d'accès à ces fichiers, vous devez mettre à jour ces chemins d'accès dans la boîte ***Préférences avancées***, accessible via le menu ***Options***, choix ***Préférences*** du traitement de texte local.

Pour une description complète du traitement de texte local de DSMS habillé, se référer au chapitre correspondant du Manuel de l'Utilisateur.

6.2.2. Installation

L'installation de DSMS habillé est identique à celle de PAW, si ce n'est que la boîte de dialogue 'Répertoires des composants de PAW' vous propose, outre les répertoires présents dans l'installation de PAW, un répertoire du programme du traitement de texte DSMS et un répertoire de Windows qui recevra le fichier TT.INI (paramétrage général du Traitement de Texte).

Pour une description détaillée de la procédure d'installation, se référer au chapitre "Installation", sous-chapitre "Procédure d'installation".

Pour parfaire l'installation du poste du développeur, consulter le chapitre "Installation", sous-chapitre "Personnalisation du poste installé". Vous y trouverez des explications sur trois fichiers :

- le fichier ENVIRON.PRM.
  Les paramètres 001, 021 et 022 ne concernent pas le développeur chargé de l'habillage de DSMS.

- le fichier PAWMAK.PRM.
  En ce qui concerne l'habillage de DSMS, le rôle de ce fichier se limite à indiquer les options de compilation et d'édition des liens pour les fichiers produits par PAWLIS.EXE dans le répertoire CHEMIN-BASE\LISTE. La valeur HELP de OPER et les exemples ne concernent pas le développeur chargé de l'habillage de DSMS.

- le fichier PAWLIB.PRM.
  Traduction dans le langue de l'utilisateur de certains libellés affichés dans les menus ***Action*** et ***Enchaînement***.

De plus, on trouvera un certain nombre de fichiers Scripts d'exemples dans le répertoire <*racine**scripts*\SAMPLES>.

6.2.3. Réalisation des listes de valeurs externes

Pour que les valeurs saisies dans les tables DSMS soient proposées à l'utilisateur à l'appel du choix ***Aide sur les valeurs*** du menu ***Aide*** ou du menu ***Enchaînement*** (voir le Manuel de l'utilisateur PAW), il est nécessaire de créer des listes externes reprenant l'ensemble de ces valeurs.

Cette opération suppose l'obtention en local d'un fichier source formaté contenant les valeurs externes.

L'extraction des informations des tables DSMS s'effectue par le lancement de la procédure DEXH, à partir de la base DSMS sur le serveur. Pour de plus amples renseignements au sujet de la procédure DEXH, consulter le Manuel d'exploitation DSMS. Le transfert en local du fichier généré est à la charge du développeur. Le fichier source peut également être créé manuellement en local, en respectant la structure requise.

Pour procéder à la suite des opérations, consulter :

- la section "Structure des fichiers sources de valeurs externes" du sous-chapitre "Liste de valeurs externes", chapitre "Fonctions avancées". Dans le cas de DSMS, NOM-LISTE doit respecter la codification suivante :

VALEURS POUR ALIMENTER LA TABLE DSMS :	CHOIX DSMS POUR ACCEDER A CETTE TABLE	NOM-LISTE CORRESPONDANT
Attributions : fonctions	TATF	ID99TATF
Attributions : responsabilités	TATR	ID99TATR
Gravité : amélioration	TGRC	ID99TGRC
Gravité : événement	TGRE	ID99TGRE
Langues	TLA	ID99TLA
Options-module	TOP	ID99TOP
Phases	TPH	ID99TPH
Produits	TPR	ID99TPR
Régions	TRE	ID99TRE
Etat-situation : amélioration	TSTC	ID99TSTC
Etat-situation : événement	TSTE	ID99TSTE
Etat-situation : site	TSTS	ID99TSTS
Filiales-pays	TSU	ID99TSU
Types événements	TTY	ID99TTY
Utilisateurs : définition	TUD	ID99TUD

Si vous avez utilisé la procédure DEXH, votre fichier possède automatiquement une structure correcte et vous pouvez passer directement à la phase de génération en local.

- la section "Génération en local des listes de valeurs externes" ainsi que la section "Compilation et vérification des fichiers obtenus" du sous-chapitre "Liste de valeurs externes", chapitre "Fonctions Avancées".
  Les fichiers obtenus après compilation dans le répertoire CHEMIN-BASE\LISTE doivent être copiés manuellement dans le répertoire CHEMIN-LIST.

6.2.4. Configuration du clavier

Lors de la première utilisation de DSMS habillé, un fichier de configuration du clavier est automatiquement créé. Ce fichier, PAW_KBRD.PRM, est décrit en détail dans le sous-chapitre "Configuration du clavier" du chapitre "Fonctions avancées".

6.2.5. En cas de problème

En cas de problème, se reporter au chapitre "Gestion des erreurs".

7. PAW serveur DDE

Vous pouvez utiliser PAW comme serveur DDE (Dynamic Data Exchange) d'une application WINDOWS cliente. Lorsque PAW fonctionne comme serveur DDE, une application WINDOWS cliente demande à WINDOWS l'ouverture d'un dialogue DDE avec PAW. Par la suite, et ceci jusqu'à la fermeture de la liaison établie entre les deux programmes, l'application cliente peut demander (REQUEST) ou envoyer (POKE) des données à PAW ou lui demander l'exécution de commandes. Dans tous ces cas, l'information transitant entre les deux applications est échangée sous forme de chaînes de caractères.

Les pages suivantes contiennent une description sommaire des principes et des mécanismes du DDE. Une bonne connaissance de ces principes et mécanismes est utile pour en tirer parti.

7.1. Caractéristiques d'un dialogue DDE

Pour ouvrir un dialogue DDE, vous devez définir l'*application* et le *sujet* avec lesquels l'application cliente va dialoguer. L'objet de ce dialogue peut ensuite être soit un *article*, soit une *commande*. Ces quatre notions sont détaillées ci-dessous :

- l'*application* est le nom du serveur (valeur par défaut : PAW); ce nom est défini par l'utilisateur et enregistré dans le fichier ENVIRON.PRM en ligne 032;

- le *sujet* ("Topic" dans les documentations en anglais) est le radical du nom du fichier script de connexion utilisé et correspond donc à une des applications serveur avec laquelle l'application cliente va dialoguer; PAW autorise aussi l'utilisation comme *sujet* de la chaîne de caractères SYSTEM (standard DDE), qui vous permet d'ouvrir des dialogues DDE indépendamment de l'application serveur courante;

- un *article* ("Item" dans les documentations en anglais) correspond aux données demandées ou envoyées par l'application cliente à l'application serveur; elles représentent soit un champ d'affichage d'un écran, soit l'écran tout entier. Un champ est désigné soit par ses coordonnées, soit de façon logique comme dans le langage des scripts. Cette désignation peut être complétée par le code de l'écran (facultatif).

- une *commande* correspond à une chaîne de caractères envoyée par l'application cliente à l'application serveur; cette chaîne de caractères a pour but de déclencher une action : l'affichage d'une fenêtre, un débranchement ou une transmission par exemple.

Pour établir un dialogue avec le serveur DDE, l'application cliente utilise un message WM_DDE_INITIATE en lui associant les deux chaînes de caractères correspondant à l'*application* et au *sujet*. L'objet du dialogue peut ensuite être soit la demande ou l'envoi d'un *article* (voir le sous-chapitre "Syntaxe des articles"), soit l'envoi d'une *commande* (voir le sous-chapitre "Syntaxe des commandes").

Pour plus de détails sur l'envoi de message et l'ouverture de dialogues DDE, se reporter à la documentation technique **WINDOWS**.

7.2. Caractéristiques des liaisons DDE

Il existe trois types de liaisons DDE : les liaisons *passives*, *automatiques* et *actives*. Dans la version courante de PAW, seules les liaisons passives sont assurées.

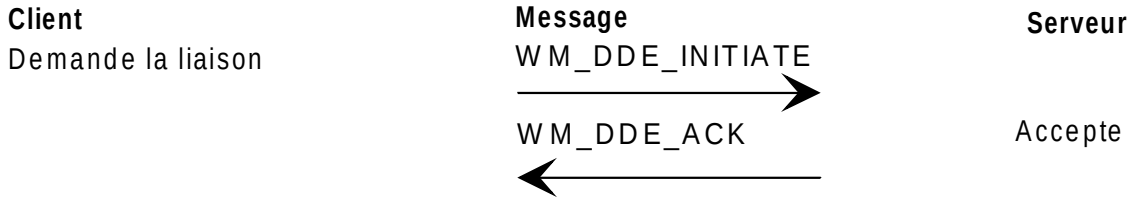
Dans une *liaison passive*, l'application cliente demande au serveur de lui fournir une donnée en lui envoyant un message WM_DDE_REQUEST. Le serveur répond par un WM_DDE_DATA. Contrairement aux liaisons automatiques et actives, lorsque la donnée est mise à jour sur le serveur, celle-ci n'est pas renvoyée au client qui doit donc la redemander explicitement.

SEQUENCES D'UNE LIAISON PASSIVE

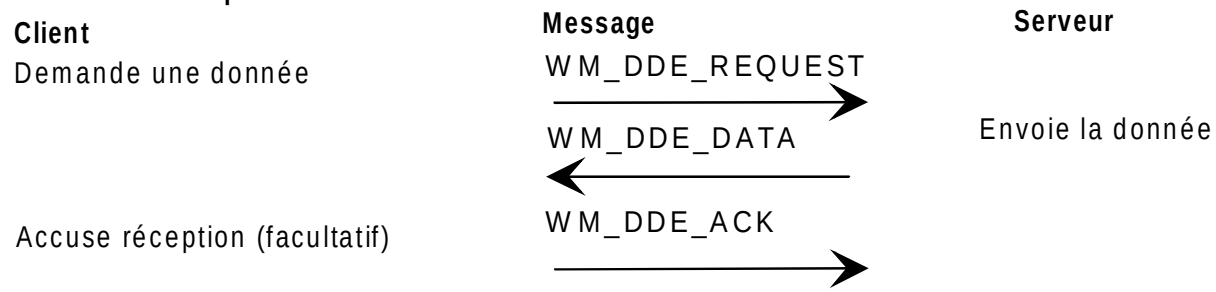
Les trois principales séquences d'une liaison DDE passive sont l'*ouverture du dialogue*, la *demande et la réception de la valeur* et la *fermeture du dialogue*.

Les échanges se déroulent de la façon suivante :

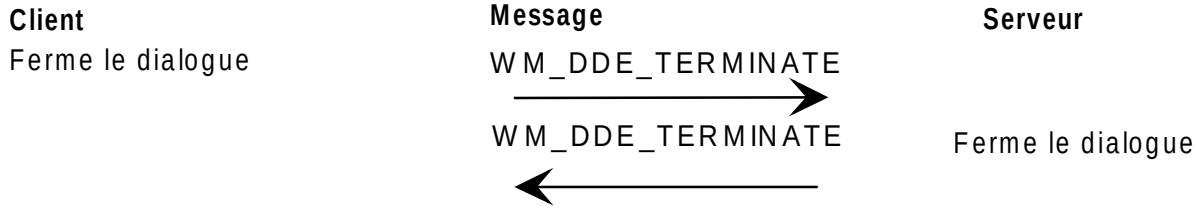
Ouverture du dialogue :



Demande et réception d'une donnée



Fermeture du dialogue



7.3. Syntaxe des articles

Un article est désigné par une expression composée de deux groupes d'éléments :

- un premier groupe d'éléments (séparés par des points) qui indiquent le type et éventuellement l'écran et le code de la donnée concernée,
- un deuxième groupe d'éléments séparés par des virgules et placés entre crochets et qui indiquent le rang ou la position de la donnée.

Les articles disponibles sont les suivants :

szEcran.FIELD [iLigne ,iColonne ,iLongueur]	Champ (FIELD) appartenant à l'écran szEcran et commençant à la position iLigne-iColonne.
szEcran.DE.szCodRub.szType [iRang]	Rubrique (DE) appartenant à l'écran szEcran, de code szCodRub, de type szType (cf. types de rubriques dans les scripts) et de rang iRang.
szEcran.ACTIONCODE [iRang]	Rubrique code action avec le iRang différent de 0 ou 1 si nécessaire.
szEcran.OPERCODE	Rubrique code opération.
szEcran.ERRORCODE [iRang]	Rubrique code d'erreur avec le iRang différent de 0 ou de 1 si nécessaire. Cette rubrique n'est pas utililisable avec le message WM_DDE_POKE.
szEcran.SCREEN	Renvoie la totalité de l'écran.
szEcran.SCREENID	Code de l'écran.
szEcran.GETCURPOS	Renvoie la position du curseur sous la forme de coordonnées: [iLigne, iColonne].

Dans tous les exemples ci-dessus, szEcran est facultatif. Il permet d'ajouter un niveau de contrôle supplémentaire sur le code écran.

7.4. Syntaxe des commandes

Une commande se compose d'un verbe, éventuellement associé à un code écran et accompagné d'un opérande.

Par exemple, szEcran.SEND ("PF01") demande la transmission sur l'écran donné de la touche fonction 1.

Les commandes disponibles sont :

szEcran.SCREENBRANCH(szOper)	Demande de débranchement d'écran. szEcran est le code de l'écran de départ. szOper est la valeur du code opération.
szEcran.SEND(szTouche)	Transmission avec le code mnémonique de la touche (cf. les scripts) dans szTouche.
szEcran.SCRIPT(szCodFic)	Demande d'exécution de script.
szEcran.SHOW(szShow)	Commande l'affichage de la fenêtre **PAW**. Le paramètre szShow prend les mêmes valeurs que le paramètre correspondant SCRSHOW des scripts.
szEcran.STATE(szEtat)	Ordre d'activation ou d'inactivation de la fenêtre **PAW**. szEtat peut prendre les valeurs NORMAL ou SILENT. En mode SILENT, **PAW** exécute toutes les tâches du mode normal mais n'envoie pas de messages d'erreur et n'exécute pas les scripts automatiques d'écran. Ceux-ci peuvent être déclenchés à la demande par l'ordre script.
szEcran.HELP(szAide)	Déclenche l'affichage de la fenêtre d'aide de **WINDOWS**. Suivant la valeur de szAide, on obtient : H_ONPAW : Aide sur paw; H_INDEX : Index de l'aide; H_ONHELP : Aide sur l'aide; *CodRub* : Aide sur la rubrique *CodRub*; *Blanc* : Aide sur l'écran courant (aide étendue).
szEcran.SETCURPOS (iLigne ,iColonne)	Positionne le curseur aux coordonnées définies par le couple (iLigne , iColonne).

7.5. Exemple d'utilisation de PAW comme serveur DDE à partir d'une application VisualBasic

Les exemples de code VisualBasic sont en italiques. Texte1 est le nom de la fenêtre "edit" de l'application VisualBasic qui est le support de la communication DDE.

1. Ouverture d'un dialogue DDE

```
ConstMANUAL = 2
Dim Chaine, Item, Ecran

Texte1.LinkTimeout = 1000
If (Texte1.LinkMode = 0) Then
  Texte1.LinkTimeout = 200
  Texte1.LinkTopic = "PAW\NPSF"
```

Ouverture d'un dialogue avec l'application PAW sur le sujet NPSF (radical du nom du script de connexion)

```
  Texte1.LinkMode = MANUAL
End if
```

2. Lecture des champs d'une répétitive de l'écran courant (pas de code d'écran)

```
Ecran = ""
For I = 1 à 18
```

Boucle de récupération des valeurs de la rubrique répétitive IDATM. La chaîne envoyée à chaque request est de la forme :
.DE.IDATM.PRPTEC[i]
où i est le rang dans la répétitive.

```
  Texte1.LinkItem = Ecran & ".DE.IDATM.PROTEC["&Str(I) & "]"
  Texte1.LinkRequest

  If (Texte1.Text = "   ")Then
        Exit For
  End if
```

3. Envoi dans le champ opération de l'écran courant d'une valeur saisie dans l'application cliente (boîte edit Texte2).

Texte1.Text = Texte2.Text	Lecture du code à transmettre
Texte1.LinkItem = ".OPERCODE"	La rubrique cible dans PAW est le code opération
Texte1.LinkPoke	Envoi de la donnée
Texte1.LinkExecute (".SEND (ENT)")	Après alimentation, on demande la transmission

4 - Envoi d'une donnée avec contrôle sur le code d'écran

Texte1.Text = Texte2.Text
Texte1.LinkItem = "ID00E2.DE.ICHOIC.OPER[1]"

Alimentation de la rubrique ICHOIC (code opération) dans l'écran ID00E2. Le transfert n'est fait que si l'écran courant est le ID00E2.

Texte1.LinkPoke

Texte1.LinkExecute (".ID00E2.SEND (ENT) ")

Transmission si l'écran courant est le ID00E2.

8. Le langage de script

8.1. Introduction

Certaines procédures sont basées sur un processus d'échange entre l'utilisateur et le système. L'utilisateur saisit une commande au terminal et la soumet au système, qui réagit en conséquence : soit il traite la commande et attend la suite, soit il la rejette et le signale à l'utilisateur.

Ces opérations s'avérant parfois fastidieuses, un langage interprété permet de les effectuer automatiquement par l'écriture de programmes, les scripts, capables d'assurer un échange avec le système, de tester sa réponse et de réagir en conséquence.

Des exemples de scripts et un exemple de fichier CONNEX.PRM sont automatiquement copiés dans le sous-répertoire SAMPLES du répertoire des Scripts lors de l'installation de PAW.

8.2. Types de scripts

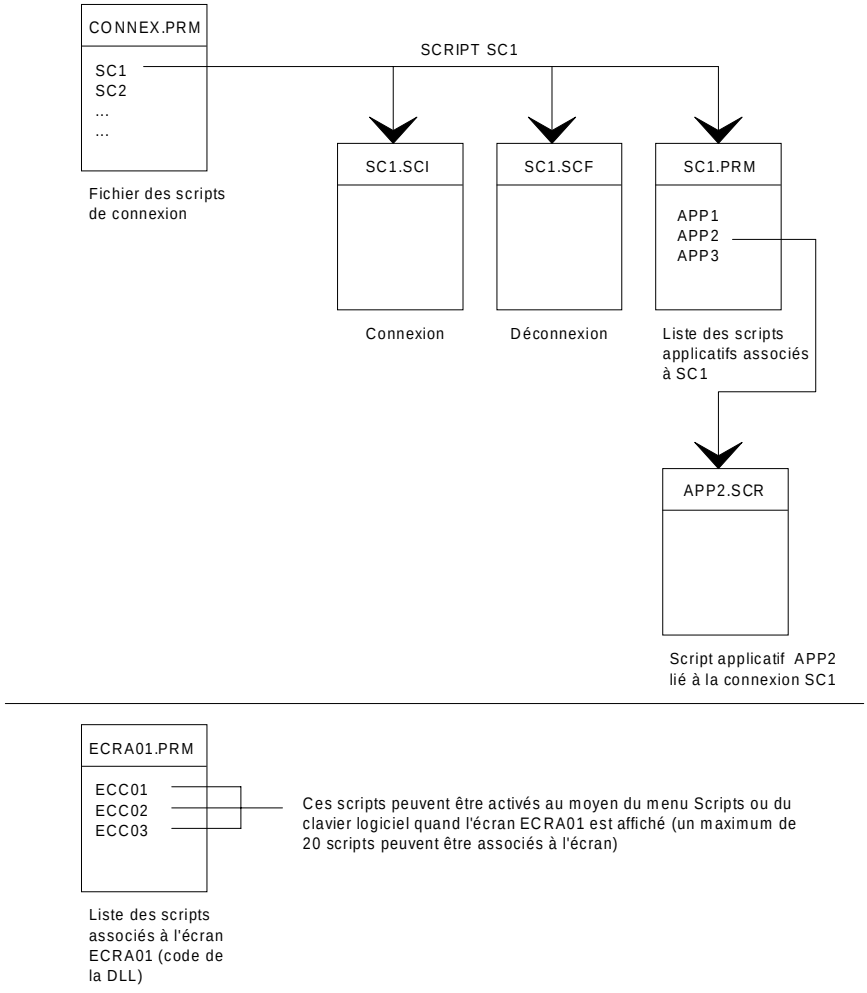
Les scripts appartiennent à deux catégories :

- Les Scripts de connexion (extension SCI) et de déconnexion (extension SCF).
  Ces scripts permettent l'exécution automatique des opérations de connexion et de déconnexion.

- Les Scripts applicatifs et d'écran (extension SCR)
  Ces scripts permettent l'exécution de séquences d'instructions au cours du travail, c'est-à-dire entre la connexion et la déconnexion.
  Les scripts applicatifs pourront être activés à tout moment pendant le travail, soit au moyen du menu Scripts, soit au moyen du clavier logiciel.
  Les scripts d'écran ne seront visibles (et donc activables) que lorsque l'écran auquel ils sont rattachés est affiché.

REMARQUES :
Le nombre maximum de scripts activables pouvant être associés à un écran est de 50.
L'installation de PAW vous permet de disposer d'exemples de scripts de (dé)connexion. Ces scripts sont automatiquement rangés dans le sous-répertoire SAMPLES.

Le schéma ci-contre illustre les différents types de scripts.



8.3. Mise en oeuvre des scripts

8.3.1. Paramétrage

Tous les fichiers nécessaires à l'utilisation des scripts de PAW sont rassemblés dans un répertoire particulier (ligne 030 du fichier ENVIRON.PRM).

8.3.1.1. Structure des lignes des listes de scripts

Les listes de scripts doivent figurer dans des fichiers d'extension .PRM. Chaque ligne de la liste doit respecter la structure suivante :

- un nom de fichier, à partir de la colonne 1 et sur 8 caractères au maximum,
- un témoin d'exécution automatique des scripts, entre les signes <>. Si ce témoin est <1>, l'exécution du script précède l'affichage de l'écran. Un seul script par écran peut comporter cette particularité. Ce témoin est facultatif.
- le numéro de l'icône associée entre crochets (voir la section "Association d'une icône à un script" dans le chapitre "Fonctions avancées", sous-chapitre "Clavier Logiciel").
- un libellé, qui débute au premier caractère non blanc, et peut s'étendre jusqu'à la fin de la ligne.

Si les lignes ne respectent pas ce format, elles ne seront pas prises en compte.

8.3.1.2. Liste des scripts de connexion

La liste des scripts de connexion est contenue dans le fichier CONNEX.PRM. Lors du lancement de PAW, une boîte de dialogue présente la liste des connexions disponibles. Le nom de fichier est le radical des deux fichiers de script associés dont les extensions sont .SCI pour la connexion et .SCF pour la déconnexion.

Le fichier CONNEX.PRM contient une ligne par application prise en compte, comme dans l'exemple suivant :

```
DSMS      DSMS d'exploitation
PAC150    PACBASE version 1.5
CICST     CICS de test
APPLI     Application exemple
```

8.3.1.3. Liste des scripts applicatifs et d'écran

Prenons l'exemple de l'application nommée APPLI. La liste des scripts liés à cette application se trouve dans le fichier APPLI.PRM. La structure de ce fichier est comparable à celle de CONNEX.PRM, par exemple :

AFFICD	Affichage d'un dossier
CREATD	Création d'un dossier

L'ensemble des scripts est activable pendant toute la connexion, c'est-à-dire entre l'exécution de APPLI.SCI et de APPLI.SCF. Les scripts correspondant aux lignes du fichier APPLI.PRM sont AFFICD.SCR et CREATD.SCR.

Les scripts d'écran sont référencés dans un fichier d'extension PRM dont le radical est celui du fichier DLL d'habillage d'écran. L'ensemble des scripts liés à l'écran DO0000 sont référencés dans le fichier DO0000.PRM sous la forme suivante :

SEARCH <1>	Recherche d'un lot technique
INITDO	Initialisation de l'écran

8.3.2. Scripts et DDE

Les scripts permettent d'établir des dialogues avec un serveur DDE grâce aux commandes suivantes :

- Ouverture d'un dialogue avec un serveur sur un sujet donné (DDEINIT),
- Demande ou fourniture de données au serveur (DDEREQUEST et DDEPOKE),
- Demande de fermeture de la conversation (DDETERMINATE),
- Modification de l'affichage de la fenêtre du serveur DDE.

8.4. Structure d'un script

La structure d'un script est :

```
PROG            Nom_du_script
déclaration(s)
BEGIN
instruction(s)
END
```

La première ligne du script doit avoir la structure suivante

```
PROG            nom_de_script
```

Elle permet d'indiquer le nom du programme.

Le mot-clé PROG peut être écrit en minuscules ou en majuscules. Le nom du programme est une suite de lettres et de chiffres commençant par une lettre, de 16 caractères au maximum. Il ne doit pas être un mot réservé. (La liste des mots réservés est donnée ci-dessous.)

Aucune autre information ne peut figurer sur cette ligne.

8.5. Mots réservés

Les mots-clés et les noms de fonctions sont réservés. Ils ne peuvent être utilisés comme noms de programmes et de variables ou comme étiquettes.

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

Les mots réservés suivants sont les noms des fonctions prédéfinies. Ils doivent être entrés en majuscules :

CONCAT	ERROR	READ
CPYSCR	EXIT	READC
CURPOS	FCLOSE	RPARAM
DATAELGET	FOPEN	SCREENID
DATAELGETAT	FREAD	SCRSHOW
DATAELGETPOS	FWRITE	SEARCH
DATAELSET	GETPROFSTR	SEND
DDEEXEC	GETSGLOBAL	SETPROFSTR
DDEINIT	INPUT	SETSGLOBAL
DDEPOKE	INTSTR	STREXTRACT
DDEREQ	OUTPUT	WRITE
DDESHOW	PAUSE	WRITEC
DDETERM	PGMEXEC	WRITEM
DISPLAY		

8.6. Déclarations

La partie déclarative du programme permet de définir les variables et étiquettes utilisées dans les instructions. Elle se situe entre l'en-tête et le corps du programme (si aucune variable n'est utilisée, cette partie peut être supprimée).

- déclaration de variable :

```
Type_de_variable    Nom_de_variable
```

- déclaration d'étiquette :

```
$label
```

8.6.1. Types de variables

Il existe trois types de variables : entier (noté integer), chaîne (notée string), booléen (noté boolean).

Une déclaration de variable s'écrit donc:

```
            STRING      nom_de_variable
ou          INTEGER     nom_de_variable
ou          BOOLEAN     nom_de_variable
```

Plusieurs variables du même type peuvent être déclarées sur une même ligne:

```
            STRING      nom1 nom2
```

est équivalent à:

```
            STRING      nom1
            STRING      nom2
```

8.6.2. Noms de variables

Un nom de variable est une suite de caractères alphanumériques, commençant par une lettre. La longueur en est limitée à 16 caractères. Toute variable utilisée dans le programme doit impérativement être déclarée. Deux variables ne peuvent pas avoir le même nom. Un nom de variable ne doit être ni un mot-clé, ni le nom d'une fonction (voir ci-dessus la liste des mots réservés).

Les noms de variable doivent être suivis d'un caractère blanc ou d'un autre caractère non alphanumérique. Une déclaration se termine impérativement par un retour à la ligne.

8.6.3. Valeurs des variables

L'indication de la valeur d'une variable s'écrit :

```
nom_de_variable=Valeur
```

Les entiers sont compris entre *(-maxint)* et *(maxint -1)*, *maxint* dépendant de la machine utilisée ($2^{15} = 32768$ sur IBM PC). Ils sont manipulés par les opérateurs décrits dans la section INSTRUCTIONS.

Les chaînes de caractères ont une longueur maximum de 256 caractères. Elles sont entrées entre guillemets. Elles peuvent comprendre des blancs. Lorsque l'un des caractères de la chaîne est le guillemet lui-même, il doit être doublé.

Lorsqu'une chaîne ne tient pas sur une seule ligne, il est nécessaire d'insérer un "&", avant le retour à la ligne. La suite de la chaîne commence à la colonne 1 de la ligne suivante. Entre le caractère de continuation et le retour à la ligne, vous ne pouvez mettre que des blancs ou des marques de tabulation, sinon le "&" est considéré comme un caractère faisant partie de la chaîne.

Exemples:

```
STRING sEXEMPLE

sEXEMPLE="Il dit ""Bonjour""."
Le contenu de la chaîne est :
Il dit "Bonjour".

sEXEMPLE="Cette chaîne se continue&
sur la ligne suivante."
Le contenu de la chaîne est :
Cette chaîne continue sur la ligne suivante.

sEXEMPLE="Cette chaîne contient un &(et commercial)."
Le contenu de la chaîne est :
Cette chaîne contient un &(et commercial).
```

Les booléens ne peuvent prendre que deux valeurs, notées *true* pour vrai et *false* pour faux. Ils peuvent être combinés avec des opérateurs logiques (voir ci-dessous). Ce sont les booléens qui entrent en compte dans toutes les structures conditionnelles.

8.6.3.1. Syntaxes du test d'un booléen

Exemple: prog TEST01SCR:

Le script suivant montre 5 manières différentes de tester un booléen.

```
boolean bBool
string stVrai
string stFaux

begin
  bBool = true
  stVrai = "bBool est VRAI"
  stFaux = " bBool est FAUX"

:       Syntaxe 1       :
        if ( BBool )
                begin
                ERROR ( "Syntaxe 1" ,"stVrai" ,1 )
                end
        else
                begin
                ERROR ( "Syntaxe 1" ,"stFaux" ,1 )
                end

:       Syntaxe 2       :
        if ( BBool == true )
                begin
                ERROR ( "Syntaxe 2" ,"stVrai" ,1 )
                end
        else
                begin
                ERROR ( "Syntaxe 2" ,"stFaux" ,1 )
                end

:       Syntaxe 3       :
        if ( NOT BBool )
                begin
                ERROR ( "Syntaxe 3" ,"stFaux" ,1 )
                end
        else
                begin
                ERROR ( "Syntaxe 3" ,"stVrai" ,1 )
                end

:       Syntaxe 4       :
        if ( NOT BBool == false )
                begin
                ERROR ( "Syntaxe 4" ,"stVrai" ,1 )
                end
        else
                begin
                ERROR ( "Syntaxe 4" ,"stFaux" ,1 )
                end
```

```
:       Syntaxe 5       :
        if ( NOT (BBool == true) )
                begin
                ERROR ( "Syntaxe 6" ,"stFaux" ,1 )
                end
        else
                begin
                ERROR ( "Syntaxe 6" ,"stVrai" ,1 )
                end
```

8.6.4. Etiquettes

Les étiquettes utilisées dans les instructions de branchement (GOTO $XXX) doivent également être déclarées. Le nom d'une étiquette est constitué de chiffres, 3 au maximum.

Déclaration d'étiquette :

```
$nom_d'étiquette
```

Aucune autre information ne peut figurer sur la ligne. Toutes les étiquettes du programme doivent avoir des noms distincts.

8.6.5. Commentaires

Les commentaires, placés entre ":", ne sont pas interprétés. Ils peuvent être insérés partout, sauf dans une chaîne de caractères. Il est possible d'écrire plusieurs lignes de commentaires en plaçant un ":" au début de la première ligne, et un second à la fin de la dernière ligne. Les instructions incluses dans un commentaire sont ignorées.

Exemples :

```
PROG            NOM_DE_PROGRAMME
:CE PROGRAMME A UN COMMENTAIRE ETALE SUR
PLUSIEURS LIGNES :
BEGIN END

PROG NOM_DE_PROG        :CE PROGRAMME A PLUSIEURS LIGNES:
BEGIN                   :DE COMMENTAIRE DANS LA MARGE:
END

PROG NOM_DE_PROG        :CE COMMENTAIRE RECOUVRE LE MOT
BEGIN                   CLE BEGIN, IL Y AURA DONC ERREUR :
END

PROG                    NOM_DE_PROGRAMME
CHAINE                  NOM_DE_CHAINE
BEGIN                   NOM_DE_CHAINE="ARTHUR:COMMENTAIRE:"
END
```

La chaîne vaut ARTHUR:COMMENTAIRE: et non ARTHUR.

8.6.6. Espacements et retours à la ligne

Les espacements et les tabulations sont équivalents, et marquent la fin d'une suite de caractères. Dans la suite de ce chapitre, seront soulignés les cas où cette séparation est indispensable. Un mot-clé immédiatement suivi d'un caractère alphanumérique n'est pas reconnu. Par exemple, un en-tête ne peut s'écrire : PROGnom

Un retour à la ligne marque la fin d'une instruction. Les cas où le retour à la ligne est obligatoire seront indiqués. Il peut s'avérer nécessaire de continuer une instruction sur plusieurs lignes, en particulier lorsqu'une fonction a plusieurs paramètres (voir ci-après). En ce cas, il est possible d'indiquer que le retour à la ligne ne représente pas la fin de l'instruction en insérant un "&" avant le retour à la ligne. La suite de la ligne ne doit pas comprendre de caractère autre que tabulation et blanc : tout autre caractère inséré après le "&" provoque une erreur.

Exemple :

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

équivaut à :

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

Lorsque le retour à la ligne est autorisé, il est possible de laisser une ou plusieurs lignes blanches, sauf dans la structure de contrôle IF ELSE (voir plus loin) qui n'admet pas de ligne blanche avant le ELSE.

8.7. Corps du programme

Le corps du programme est délimité par le mot-clé BEGIN et le mot-clé END. Il est constitué d'une suite d'instructions et de structures, dont les syntaxes sont décrites plus loin.

8.8. Instructions

Les instructions constituant le corps du programme sont constituées d'appels de fonctions, de sauts à une autre partie du programme et d'affectations de valeurs aux expressions.

8.9. Affectation

L'affectation permet d'attribuer à une variable la valeur d'une expression. Elle s'écrit :

```
nom_de_variable = expression
```

Aucune autre information ne peut figurer sur la ligne.

L'expression peut être soit une autre variable, soit une constante de type entier, chaîne ou booléen, soit un appel de fonction, soit enfin une expression où des opérateurs sont des expressions, selon les règles édictées dans le sous-chapitre qui suit.

8.10. Expressions et Opérateurs

Les expressions simples sont du type :

- nom_de_variable
- constante entière
- constante littérale
- constante booléenne

Des opérateurs arithmétiques et logiques permettent d'obtenir des expressions complexes :

expression : expression$_A$ opérateur expression$_B$

Des opérateurs arithmétiques combinés avec des opérandes entiers permettent d'obtenir des entiers:

- expression$_A$	opposé de expression$_A$
expression$_A$ + expression$_B$	somme des expressions
expression$_A$ - expression$_B$	différence des expressions
expression$_A$ * expression$_B$	produit des expressions
expression$_A$ / expression$_B$	quotient des expressions

Des opérateurs logiques permettent d'obtenir des résultats booléens (dont la valeur est soit TRUE, soit FALSE) :

- exp$_A$ < exp$_B$ donne TRUE si exp$_A$ est inférieur à exp$_B$; exp$_A$ et exp$_B$ doivent être entiers.
- exp$_A$ > exp$_B$ donne TRUE si exp$_A$ est supérieur à exp$_B$; exp$_A$ et exp$_B$ doivent être entiers.
- exp$_A$ == exp$_B$ donne TRUE si exp$_A$ est égal à exp$_B$; exp$_A$ et exp$_B$ doivent être de même type.
- exp$_A$ <> exp$_B$ donne TRUE si exp$_A$ est différent d'exp$_B$; exp$_A$ et exp$_B$ doivent être de même type.
- exp$_A$ and exp$_B$ donne TRUE si exp$_A$ et exp$_B$ sont vrais ; exp$_A$ et exp$_B$ doivent être booléens.
- exp$_A$ or exp$_B$ donne TRUE si exp$_A$ ou exp$_B$ est vrai ; exp$_A$ et exp$_B$ doivent être booléens.
- not exp$_A$ donne TRUE si exp$_A$ est FALSE et vice versa ; exp$_A$ doit être booléen.

8.10.1. Priorité des opérateurs

Chaque opérateur est doté d'une priorité et de propriétés relatives à l'associativité. L'opération correspondant à l'opérateur de plus grande priorité est d'abord effectuée, puis on passe à l'opérateur de seconde priorité etc. Si deux opérateurs de même priorité sont adjacents, on se réfère à leur associativité. Les opérateurs sont indiqués ci-dessous dans un ordre de priorité décroissante.

-	unaire
* /	associatifs
- +	associatifs
< > <> ==	non associatifs
NOT	non associatif
AND OR	associatifs

8.10.2. Traitement des opérateurs associatifs

Pour évaluer A + B + C
on effectue intermédiaire = A + B
et le résultat est : intermédiaire + C.

Deux opérateurs non associatifs ne peuvent pas être adjacents. Ainsi le produit de A par la somme de B et de C doit s'écrire :

A*B+A*C

Les parenthèses "(" et ")" permettent de changer l'ordre d'évaluation. Une expression entre parenthèses équivaut à une variable temporaire, les parenthèses les plus profondes étant évaluées en priorité. Ainsi :

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

est équivalent à :

intermédiaire_1=B+C
intermédiaire_2=intermédiaire_1-D
intermédiaire_3=E/F
A=intermédiaire_2*intermédiaire_3

Les parenthèses permettent également de rendre les expressions plus lisibles sans modifier l'ordre d'évaluation. Leur utilisation est recommandée pour les expressions dont les opérandes sont de types différents, par exemple:

(a>0) and (a<11)

8.11. Branchements

L'instruction GOTO $XXX
permet de ne pas effectuer le programme en séquence en allant directement à l'endroit désigné par l'étiquette $XXX.

Le blanc entre "GOTO" et "$" est facultatif, et le retour à la ligne obligatoire.

Tout branchement n'est pas licite : on ne peut entrer dans un bloc d'instructions par un branchement, même si l'on peut en sortir.

8.12. Structures de contrôle

Les structures de contrôle permettent d'effectuer des choix et des itérations.

Les structures de contrôle sont :

IF condition bloc ELSE bloc	WHILE condition bloc UNTIL condition	DO bloc	FOR condition bloc

Une condition s'écrit :

```
( expression )
```

où expression a un type booléen. La condition est vraie si et seulement si ce booléen vaut TRUE.

La condition de la structure FOR est particulière ; elle est décrite ci-dessous.

La condition et le mot-clé qui la précède doivent être écrits sur la même ligne. Ils peuvent être séparés par un blanc.

Un bloc d'instructions regroupe plusieurs instructions qui sont exécutées en séquence en fonction de critères spécifiés par une structure de contrôle. Ces instructions peuvent elles-même être des structures de contrôle.

Le bloc d'instructions est précédé du mot-clé BEGIN et suivi du mot-clé END.

Les branchements peuvent se faire uniquement depuis l'intérieur d'un bloc vers l'extérieur ; les branchements depuis l'extérieur dans un bloc ne sont pas possibles.

8.13. Ecriture des expressions

Un bloc d'instructions s'écrit :

```
BEGIN        instructions    END
```

Un retour à la ligne est obligatoire après le bloc. On peut en insérer avant le bloc, après BEGIN, avant et après les instructions.

DO UNTIL

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

Le bloc est exécuté tant que la condition est fausse. Il est exécuté au moins une fois. Il faut bien entendu s'assurer que la condition évolue dans le temps vers la valeur TRUE, si l'on ne veut pas engendrer une boucle infinie.

FOR

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

La variable "nom_de_variable" doit être de type entier : elle sert de compteur. *ExpressionA* est évaluée une fois ; sa valeur doit être un entier, c'est la borne inférieure. De même, *expressionB* fournit la borne supérieure. Le compteur prend la valeur d'*ExpressionA* ; si *expressionB* lui est supérieure ou égale, le bloc est exécuté. Le compteur est incrémenté et est à nouveau comparé à la borne supérieure.

Si *expressionA* est supérieure à *expressionB*, le bloc n'est pas exécuté, et le programme reprend en séquence à la fin du bloc.

Cette structure permet d'exécuter un bloc un nombre fixé de fois (*expression*$_B$ - *expression*$_A$ + 1). Elle utilise un compteur, qui repère l'itération en cours. Ce compteur ne peut être utilisé en tant que variable dans le bloc : il ne peut ni subir d'affectation ni servir de compteur pour une boucle FOR imbriquée. S'il est utilisé par ailleurs dans le programme, sa valeur antérieure est perdue à l'entrée de la boucle FOR.

IF ELSE

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

Quand l'interpréteur rencontre cette structure, il évalue la condition, puis exécute le premier bloc si elle est vraie, le second dans le cas contraire.

La partie ELSE BEGIN instructions END est facultative. Dans ce cas, le programme poursuit immédiatement en séquence si la condition est fausse.

On peut insérer un seul retour à la ligne entre le premier END et ELSE.

WHILE

```
WHILE       condition
            BEGIN
            instructions
            END
```

Le bloc est exécuté tant que la condition est vraie. Si la condition est fausse dès la première évaluation, le bloc n'est jamais exécuté. Il faut bien entendu s'assurer que la condition évolue dans le temps vers la valeur FALSE, si l'on ne veut pas engendrer une boucle infinie.

Rupture de boucle

Le mot-clé BREAK permet de sortir de la structure FOR, WHILE ou DO UNTIL qui l'englobe. Cela permet de sortir prématurément d'une structure itérative, en général à la suite d'un test. Un BREAK en dehors de toute structure FOR, WHILE ou DO UNTIL provoque une erreur.

Le mot BREAK doit être suivi d'un retour à la ligne.

8.14. Fonctions

Les fonctions peuvent être utilisées comme instructions ou comme expressions. Certaines fonctions rendent une valeur (code retour). D'autres sont destinées à inter-agir avec l'environnement (affichage sur la sortie standard par exemple), et permettent au programme de dialoguer avec l'extérieur.

Vous trouverez des exemples de fonctions du langage de script dans les scripts d'exemples fournies à l'intallation de PAW (sous-répertoire SAMPLES).

8.14.1. Appel d'une fonction

L'appel d'une fonction est de la forme :

```
nom_de_fonction (liste_de_paramètres)
```

Pour chaque fonction, il sera précisé si elle rend une valeur ou non et le type de la valeur le cas échéant. Les fonctions qui rendent des valeurs peuvent être utilisées, soit dans une affectation, soit comme instruction, mais la valeur du code retour est alors perdue. En revanche, une fonction ne rendant pas de valeur ne peut figurer que dans une instruction.

8.14.2. Paramètres d'une fonction

Les paramètres sont de deux types : les paramètres constants et les paramètres modifiables. Un paramètre constant n'est pas modifié par la fonction. Ce peut être une expression ou une variable. Un paramètre modifiable est transformé par la fonction. Ces modifications sont indiquées dans la description des fonctions.

Les paramètres doivent être séparés par une virgule ; ils peuvent être de type entier, chaîne ou booléen. Les paramètres de type chaîne doivent comporter des double-quotes. Le nombre et le type de chacun des paramètres sont indiqués pour chaque fonction et doivent être respectés (sous peine d'erreur à l'exécution).

Remarque : Dans la description des fonctions, le mot ENTR indique que la valeur du paramètre (variable ou constant) est identique en entrée et en sortie. Le mot SORT indique que la valeur du paramètre (variable) a été modifiée en sortie. CR représente le code retour.

CONCAT

Concaténation de chaînes de caractères

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

Parm1	sort chaîne	chaîne recevant le résultat de la concaténation
ParmM à ParmX	entr chaîne	chaînes à concaténer
cr	entier	longueur de la chaîne concaténée

Si la chaîne recevant le résultat n'est pas vide, elle est également concaténée ; s'assurer que la longueur de la chaîne concaténée est bien inférieure ou égale à 256 (longueur maximum d'une chaîne).

CPYSCR

Copie de l'écran du serveur sur un fichier PC

Cette commande n'interrompt pas l'exécution du script; elle est surtout utile pour les tests et en cas d'erreur pour permettre à l'utilisateur de retrouver la cause d'erreur.
Les écrans se cumulent dans le fichier dont la destruction est à la charge de l'utilisateur.

cr = CPYSCR (Parm1)

Parm1	entr chaîne	nom complet du fichier dos sous la forme : ("U:\PATH\PREFIX.EXT")
cr	entier	1

CURPOS

Détermination de la position du curseur sur l'écran serveur

CURPOS (Parm1,Parm2)

Parm1	sort entier	numéro de ligne
Parm2	sort entier	numéro de colonne
cr	entier	1

DATAELGET

Valeur d'une rubrique dont le code, le type et le rang sont spécifiés

cr = DATAELGET (Parm1, Parm2, Parm3, Parm4)

Parm1	entr chaîne	code de la rubrique	
Parm2	entr chaîne	type de la rubrique LABEL : libellé fixe PROTEC : zone protégée modifiable ERROR : libellé d'erreur INPUT : zone saisissable OPER : code opération ACTION : code mouvement	
Parm3	entr entier	rang de la rubrique (cas d'une répétitive) (si Parm2 = 0 ou 1, la première valeur rencontrée est prise en compte)	
Parm4	sort chaîne	valeur de la rubrique	
cr	entier	1	commande exécutée
		-1	écran non reconnu
		-2	rubrique non trouvée

DATAELGETAT

Informations sur une rubrique dont la position est spécifiée

cr = DATAELGETAT (Parm1, Parm2, Parm3, Parm4, Parm5, Parm6)

Parm1	sort chaîne	code de la rubrique	
Parm2	sort chaîne	type de la rubrique : LABEL : libellé fixe PROTEC : zone protégée modifiable ERROR : libellé d'erreur INPUT : zone saisissable OPER : code opération ACTION : code mouvement	
Parm3	sort entier	rang de la rubrique (cas d'une répétitive) (si Parm2 = 0 ou 1, la première valeur rencontrée est prise en compte)	
Parm4	sort entier	longueur du champ	
Parm5	entr entier	numéro de ligne	
Parm6	entr entier	numéro de colonne	
cr	entier	1	commande exécutée
		-1	écran non reconnu
		-2	rubrique non trouvée

DATAELGETPOS

Position d'une rubrique dont le code, le type et le rang sont spécifiés

cr = DATAELGETPOS (Parm1, Parm2, Parm3, Parm4, Parm5)

Parm1	entr chaîne	code de la rubrique
Parm2	entr chaîne	type de la rubrique LABEL : libellé fixe PROTEC : zone protégée modifiable ERROR : libellé d'erreur INPUT : zone saisissable OPER : code opération ACTION : code mouvement
Parm3	entr entier	rang de la rubrique (cas d'une répétitive) (si Parm2 = 0 ou 1, la première valeur recontrée est prise en compte)
Parm4	sort entier	numéro de ligne
Parm5	sort entier	numéro de colonne

cr	entier	1	commande exécutée
		-1	écran non reconnu
		-2	rubrique non trouvée

DATAELSET

Mise à jour du champ de la rubrique dont le code et le rang sont spécifiés

cr = DATAELSET (Parm1, Parm2, Parm3)

Parm1	entr chaîne	code de la rubrique
Parm2	entr entier	rang de la rubrique (cas d'une répétitive) (si Parm2 = 0 ou 1, la première valeur recontrée est prise en compte)
Parm3	entr chaîne	valeur de mise à jour

cr	entier	1	commande exécutée
		-1	écran non reconnu
		-2	rubrique non trouvée

REMARQUE : Cette fonction doit être suivie d'un SEND ("ENT") pour que la mise à jour soit effective.

DDEEXEC

Envoi d'une commande au serveur

cr = DDEEXEC (Parm1, Parm2, Parm3)

Parm1	entr entier	identificateur du dialogue DDE obtenu grâce à DDEINIT
Parm2	entr chaîne	commande du serveur. Cette ligne doit respecter la syntaxe du langage de commande de l'application serveur
Parm3	entr entier	délai maximum (en secondes) accordé par le client pour l'exécution de la commande.

cr	entier	-1	dialogue inconnu
		0	commande non exécutée
		1	commande exécutée

DDEINIT

Initialisation d'un dialogue DDE

cr = DDEINIT (Parm1, Parm2, Parm3, Parm4)

Parm1	entr chaîne	nom de l'application fonctionnant comme serveur DDE
Parm2	entr chaîne	sujet du dialogue : TOPIC de la documentation DDE
Parm3	entr chaîne	ligne de commande à exécuter lorsque l'application avec laquelle on veut dialoguer n'est pas lancée. Dans le cas de WINWORD, le parm2 doit être identique au paramètre de lancement utilisé dans le parm3. Le parm3 peut aussi être utilisé pour lancer une autre application.
Parm4	entr chaîne	type d'affichage : SHOW pour afficher la fenêtre avec ses taille et position courantes NORMAL pour afficher la fenêtre. Si la fenêtre est en icône ou à la taille maximale, elle est affichée avec sa taille de départ HIDE pour cacher la fenêtre ICON pour mettre la fenêtre en icône MAX pour afficher la taille maximale de la fenêtre. En cas d'erreur sur la valeur de ce paramètre, le système prend SHOW.
cr	chaîne ou entier	identificateur du dialogue DDE en cas d'initialisation réussie, 0 sinon

DDEPOKE

Mise à jour forcée d'une donnée au niveau du serveur

cr = DDEPOKE (Parm1, Parm2, Parm3, Parm4, Parm5)

Parm1	entr entier	identificateur du dialogue DDE obtenu grâce à DDEINIT
Parm2	entr chaîne	objet à mettre à jour. (ex : une cellule d'EXCEL ou un signet de WINWORD)
Parm3	entr chaîne	valeur de mise à jour
Parm4	entr chaîne	Format de la donnée (format : CF_TEXT)
Parm5	entr entier	délai maximum (en secondes) accordé par le client pour l'exécution de la commande

cr	entier	-1	dialogue inconnu
		0	commande non exécutée
		1	commande exécutée

DDEREQ

Demande d'une donnée au serveur

cr = DDEREQ (Parm1, Parm2, Parm3, Parm4, Parm5)

Parm1	entr entier	identificateur du dialogue DDE obtenu grâce à DDEINIT
Parm2	entr chaîne	objet dont on désire la valeur (ex : une cellule d'EXCEL)

Parm3	sort chaîne	chaîne de caractères contenant la valeur demandée
Parm4	entr chaîne	Format de la donnée (format : CF_TEXT)
Parm5	entr entier	délai maximum (en secondes) accordé par le client pour l'exécution de la commande

cr	entier	-1	dialogue inconnu
		0	commande non exécutée
		1	commande exécutée

DDESHOW

Modification de l'affichage de la fenêtre du serveur

cr = DDESHOW (Parm1, Parm2)

Parm1	entr entier	identificateur du dialogue DDE obtenu grâce à DDEINIT
Parm2	entr chaîne	type d'affichage : SHOW pour afficher la fenêtre avec ses taille et position courantes NORMAL pour afficher la fenêtre. Si la fenêtre est en icône ou à la taille maximale, elle est affichée avec sa taille de départ HIDE pour cacher la fenêtre ICON pour mettre la fenêtre en icône MAX pour afficher la taille maximale de la fenêtre. En cas d'erreur sur la valeur de ce paramètre, le système prend SHOW.

cr	entier	-1	dialogue inconnu
		0	commande non exécutée
		1	commande exécutée

DDETERM

Fermeture du dialogue DDE

cr = DDETERM (Parm1)

Parm1	entr entier	identificateur du dialogue DDE obtenu grâce à DDEINIT. Cette commande est **obligatoire**.

cr	entier	-1	dialogue inconnu
		0	commande non exécutée
		1	commande exécutée

DISPLAY

Affichage de l'écran serveur

cr = DISPLAY()

Interrompt l'exécution du script et affiche dans une fenêtre l'écran du serveur. A utiliser en phase de test ou en cas d'erreur, afin que l'utilisateur puisse vérifier la cause de l'erreur. L'écran affiché n'est pas saisissable.

Aucun Paramètre

cr	entier	1

ERROR

Message d'erreur dans une boîte de dialogue

cr = ERROR (Parm1, Parm2, Parm3)

Parm1	entr chaîne	en-tête de la boîte
Parm2	entr chaîne	message d'erreur
Parm3	entr entier	boutons accessibles à l'utilisateur :

Windows français	Windows anglais
1 OK	1 OK
2 OK et Annuler	2 OK Cancel
3 Abandon Reprise Ignore	3 Abort Retry Ignore
4 Oui Non Annuler	4 Yes No Cancel
5 Oui Non	5 Yes No
6 Reprise Annuler	6 Retry Cancel

cr entier valeur du bouton pressé par l'utilisateur :

Windows français	Windows anglais
1 OK	1 OK
2 Annuler	2 Cancel
3 Abandon	3 Abort
4 Reprise	4 Retry
5 Ignore	5 Ignore
6 Oui	6 Yes
7 Non	7 No

EXIT

Sortie de l'exécution du script

EXIT (Parm1)

Parm1	entr entier	valeur différente de 0 et de 1

FCLOSE

Fermeture d'un fichier

cr = FCLOSE (Parm1)

Parm1	entr chaîne	nom du fichier, éventuellement avec le chemin complet
cr	entier	index de fermeture du fichier

FEXIST

Test de l’existence d'un fichier

cr = FEXIST (Parm1)

Parm1	entr chaîne	nom du fichier, éventuellement avec le chemin complet
cr	entier	1 si le fichier existe

FOPEN

Ouverture d'un fichier

cr = FOPEN (Parm1, Parm2)

Parm1	entr chaîne	nom du fichier, éventuellement avec le chemin complet
Parm2	entr chaîne	mode d'ouverture ('r' ou 'R' pour lecture, 'w' ou 'W' pour écriture et 'a' ou 'A' pour ajout)
cr	entier	index d'ouverture du fichier (supérieur à zéro) si l'ouverture s'effectue correctement. Code d'erreur sinon. -1 : mode d'ouverture incorrect -2 : trop de fichiers -3 : autre erreur

FREAD

Lecture d'un fichier

cr = FREAD (Parm1, Parm2)

Parm1	entr chaîne	nom du fichier, éventuellement avec le chemin complet	
Parm2	sort chaîne	chaîne réceptrice de la zone lue. Le délimiteur est le caractère CR ou la fin de fichier.	
cr	entier	-2	erreur (cf message d'erreur)
		-1	fin de fichier atteinte
		>= 0	nombre de caractères lus (si cr = 0, la ligne lue est vide)

FWRITE

Ecriture d'un fichier

cr = FWRITE (Parm1, Parm2, Parm3)

Parm1	entr chaîne	nom du fichier, éventuellement avec le chemin complet
Parm2	entr chaîne	chaîne de caractères à écrire
Parm3	entr entier	nombre de retours à la ligne avant l'écriture
cr	entier	index d'ouverture du fichier

GETPROFSTR

Alimentation avec un des éléments du fichier ENVIRON.PRM

cr = GETPROFSTR (Parm1, Parm2, Parm3)

Parm1	entr chaîne	section du fichier ENVIRON.PRM : PAW, COMM, PATHS, CUSTOM, SYSTEM
Parm2	entr chaîne	élément du fichier ENVIRON.PRM. Cet élément doit être précédé de la section à laquelle il appartient. Le numéro entre parenthèses est le numéro de la ligne dans le fichier ENVIRON.PRM. Pour la section **PAW** : Version (001), UserLang : langue du poste utilisateur (001), DevLang : langue du poste développeur (001), Target System : système cible (007), LearningMode : mode d'apprentissage (023), WorkStation : type de poste (024). Pour la section **COMM** : ProgramName : nom du gestionnaire de communication (002), ParamName : fichier de paramètres associé (002). Pour la section **PATHS** : Exe : chemin des fichiers .exe (003), User : chemin de l'utilisateur (004), ListDlls : chemin des listes de valeurs (005), Developer : chemin du développeur (006), Communication : répertoire des fichiers de GSCOM (027), Parameters : chemin des fichiers de paramétrage (028), Scripts : chemin des scripts (030), TText : chemin du traitement de texte DSMS (034). Pour la section **CUSTOM** : IconDll : fichier des icônes (020). Pour la section **SYSTEM** : PollingRate : délai d'interrogation (020), ScreenAdjust : ajustement automatique des écrans (021), ExternalNames : noms externes (022), ScriptWindow : affichage de la fenêtre script en mode test (031).
Parm3	sort chaîne	chaîne qui recevra l'élément spécifié.
cr	-1 0	échec succès

GETSGLOBAL

Interrogation d'une variable globale.

cr = GETSGLOBAL (Parm1, Parm2)

Parm1	entr chaîne	nom de la variable globale à interroger.
Parm2	entr chaîne	valeur de cette variable.
cr	1	commande exécutée (variable trouvée)

INPUT

Création d'une boîte de dialogue

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

Cette fonction a un nombre variable de paramètres :

Parm1	entr chaîne	titre de la boîte de dialogue

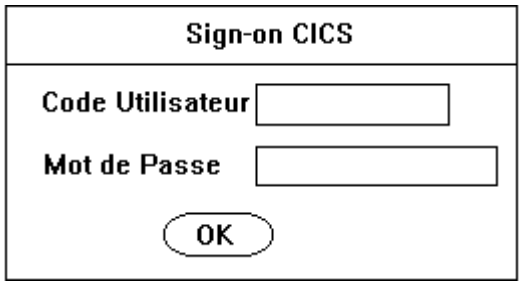
Les paramètres se regroupent ensuite par quatre, chaque groupe correspondant à une ligne de dialogue dans la fenêtre; le nombre de lignes n'est pas limité.

ParmN	entr chaîne	libellé de la ligne
ParmN+1	sort chaîne	variable mémorisant la valeur saisie par l'utilisateur
ParmN+2	entr entier	nombre de caractères de la zone saisissable
ParmN+3	entr entier	affichage des caractères saisis : 1 OUI 2 NON (Mot de Passe)
cr	entier	1 OK

Exemple :

INPUT("Sign-on CICS","Code Utilisateur",UTI,6,1,"Mot de Passe",PAS,8,2)

Cette commande permet de construire la boîte de dialogue suivante:



Le code utilisateur (UTI) visible, est sur 6 caractères, le mot de passe (PAS) invisible est sur 8 caractères. La chaîne correspondant au titre ("Sign-on CICS") est tronquée à 50 caractères.

INTSTR

Conversion d'un entier en une chaîne

cr = INTSTR (Parm1, Parm2)

Parm1	sort chaîne	chaîne résultat
Parm2	entr entier	entier à convertir
cr	entier	longueur de la chaîne résultat

OUTPUT

Message devant apparaître à l'utilisateur

cr = OUTPUT (Parm1, Parm2)

Parm1	entr chaîne	message à afficher
Parm2	entr entier	type d'affichage
0 ou >6		le message s'intègre dans la boîte deconnexion, permet de matérialiser les différentes phases de la connexion.
1 à 6		boîte de message en surimpression sur la boîte de connexion ; utiliser plutôt l'instruction ERROR (voir cette fonction pour la signification des valeurs)
cr	entier	1 si la valeur du paramètre 2 est 0 ; voir les codes retour de ERROR dans les autres cas.

PAUSE

Temporisation

cr = PAUSE (Parm1)

Parm1	entr entier	temps en secondes
cr	entier	1

PGMEXEC

Lancement d'une application WINDOWS

cr = PGMEXEC (Parm1, Parm2)

Parm1	entr chaîne	ligne de commande contenant le nom du programme à lancer et éventuellement un ou plusieurs paramètres
Parm2	entr chaîne	type d'affichage : SHOW :pour afficher la fenêtre avec ses taille et position courantes NORMAL pour afficher la fenêtre. Si la fenêtre est en icône ou à la taille maximale, elle est affichée avec sa taille de départ HIDE pour cacher la fenêtre ICON pour mettre la fenêtre en icône MAX pour afficher la taille maximale de la fenêtre. En cas d'erreur sur la valeur de ce paramètre, le système prend SHOW.

cr	entier	0	commande non exécutée
		1	commande exécutée

READ

Lecture d'une chaîne

cr = READ (Parm1, Parm2, Parm3, Parm4)

Parm1	entr entier	ligne
Parm2	entr entier	colonne
Parm3	sort chaîne	chaîne recevant le résultat de la lecture
Parm4	entr entier	longueur de la zone de lecture

cr	entier	1	lecture réussie

READC

Lecture d'une chaîne sur l'écran à la position du curseur

cr = READC (Parm1,Parm2)

Parm1	sort chaîne	chaîne recevant le résultat de la lecture
Parm2	entr entier	longueur de la zone de lecture

cr	entier	1	lecture réussie

RPARAM

Lecture, durant l'exécution du script, des paramètres indiqués pour la connexion à PACBASE

cr = RPARAM (Parm1, Parm2)

Parm1	entr entier	type de paramètre
Parm2	sort chaîne	chaîne recevant le paramètre

cr	entier	1	OK
		2	Valeur de Parm1 erronée

Type de paramètre

0	code utilisateur
1	mot de passe
2	code base
3	code bibliothèque
4	numéro de session (session courante : 9999)
5	type de session (session historisée : H / session test : T)
6	code produit DSMS
7	numéro d'amélioration DSMS

SCREENID

Alimentation d'une chaîne de caractères avec le code de la DLL d'écran

SCREENID (Parm1)

Parm1	sort chaîne	chaîne réceptrice du code de la DLL

SCRSHOW

Affichage d'un écran PAW préalablement masqué

cr = SCRSHOW (Parm1)

Parm1	entr chaîne	type d'affichage : SHOW pour afficher la fenêtre avec ses taille et position courantes NORMAL pour afficher la fenêtre. Si la fenêtre est en icône ou à la taille maximale, elle est affichée avec sa taille de départ HIDE pour cacher la fenêtre ICON pour mettre la fenêtre en icône MAX pour afficher la taille maximale de la fenêtre. En cas d'erreur sur la valeur de ce paramètre, le système prend SHOW.	
cr	entier	0	commande non exécutée
		1	commande exécutée

SEARCH

Recherche d'une chaîne de caractères sur l'écran

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

Parm1	sort entier	ligne de début de recherche (toutes lignes : valeur 0)	
Parm2	sort entier	colonne de début de recherche (toutes colonnes : valeur 0)	
Parm3	entr chaîne	chaîne à rechercher	
Parm4	entr entier	longueur de la chaîne	
Parm5	entr entier	temps maximum de recherche en secondes.	
cr	entier	0	chaîne non trouvée
		1	chaîne trouvée

REMARQUE :

Parm4 n'est pas utilisé systématiquement :

- si le paramètre longueur vaut zéro ou s'il est supérieur à la longueur de la chaîne à rechercher, il est ignoré;
- si le paramètre longueur est inférieur à la longueur de la chaîne à rechercher, la fonction recherche la sous-chaîne de longueur égale au paramètre longueur;
- la valeur 0 est ignorée.

EXEMPLE:

```
SEARCH (ln, cl , "AAAA"  , 12 , 10 ) recherche "AAAA"
SEARCH (ln, cl , "AAAAAAAA" , 4, 10) recherche "AAAA".
```

SEND

Envoi d'une touche

cr = SEND (Parm1)

Parm1		entr chaîne	touche à envoyer
cr	entier	1	envoi normal
		2	touche inconnue
		3	pas de réponse du site central

TOUCHES TOUS MATERIELS

ENT	enter
GCHE	déplacement curseur vers la gauche
DRTE	déplacement curseur vers la droite
HAUT	déplacement curseur vers le haut
BAS	déplacement curseur vers le bas
TAR	tabulation arrière (écran formaté)
TAV	tabulation avant (écran formaté)
PF01 à PF24	Touches fonctions

PROTOCOLE 3270 SEULEMENT

ATTN	attn
ATTR	identifiants
CLS	clear
CURS	cursor select
DEL	supprime un caractère
DUP	dup
EEOF	erase EOF
ERA	erase
HOME	début de ligne
PA1	AP1
PA2	AP2
PA3	AP3
PRN	print
REST	restaure
RSET	reset
SYST	appel système

PROTOCOLE VIP-QUESTAR SEULEMENT

CLS	effacement local d'écran + transmission
FLC	effacement local d'écran
HOME	premier caractère saisissable de l’écran

SETPROFSTR

Modification de l'environnement de travail (Parm1 et Parm2) avec le contenu de Parm3.

Attention : le fichier ENVIRON.PRM n'est pas modifié. Seules les valeurs en mémoire sont affectées.

cr = SETPROFSTR (Parm1, Parm2, Parm3)

Parm1	sort chaîne	section du fichier ENVIRON.PRM : PAW ou PATHS.
Parm2	sort chaîne	élément du fichier ENVIRON.PRM. Cet élément doit être précédé de la section à laquelle il appartient. Le numéro entre parenthèses est le numéro de la ligne dans le fichier ENVIRON.PRM. Pour la section **PAW** : UserLang : langue de l'utilisateur (001). Pour la section **PATHS** : ListDlls : chemin des listes de valeurs (005), Scripts : chemin des scripts (030).
Parm3	entr chaîne	chaîne contenant une valeur mettant à jour un élément du fichier ENVIRON.PRM.
cr	-1 0	échec succès

SETSGLOBAL

Mémorisation d’une variable globale.

cr = SETSGLOBAL (Parm1, Parm2)

Parm1	sort chaîne	nom de la variable globale à positionner ou à modifier (si elle existe déjà).
Parm2	entr chaîne	valeur à attribuer à cette variable.
cr	1	commande exécutée

STREXTRACT

Extraction d'un segment de chaîne

cr = STREXTRACT (sOrigine, iDébut, iLongueur, sCible)

cr	entier	longueur de la chaîne extraite

WRITE

Ecriture d'une chaîne à l'écran

cr = WRITE (Parm1, Parm2, Parm3 , Parm4)

Parm1	entr entier	ligne
Parm2	entr entier	colonne
Parm3	entr chaîne	chaîne à écrire
Parm4	entr entier	modification du MDT (mode 3270) affichage de la chaîne en mode test
0	pas de modification du MDT, chaîne visible	
1	modification du MDT, chaîne visible	
2	pas de modification du MDT, chaîne masquée	
3	modification du MDT, chaîne masquée	

cr	entier	1	écriture réussie
		2	erreur écriture

REMARQUE : Cette fonction doit être suivie d'un SEND ("ENT") pour que la mise à jour soit effective.

WRITEC

Ecriture d'une chaîne à l'écran à la position du curseur

cr = WRITEC (Parm1, Parm2)

Parm1	entr chaîne	chaîne à écrire
Parm2	entr entier	modification du MDT (mode 3270) affichage de la chaîne en mode test
0	pas de modification du MDT, chaîne visible	
1	modification du MDT, chaîne visible	
2	pas de modification du MDT, chaîne masquée	
3	modification du MDT, chaîne masquée	

cr	entier	1	écriture réussie
		2	erreur écriture

(Si le curseur n'est pas sur une zone saisissable, l'écriture échoue.)

REMARQUE : Cette fonction doit être suivie d'un SEND ("ENT") pour que la mise à jour soit effective.

WRITEM

Ecriture d'une chaîne en mode non formaté - BULL seulement

cr = WRITEM (Parm1,Parm2)

Parm1	entr chaîne	chaîne à écrire	
Parm2	entr entier	en mode test :	0 = chaîne visible 2 = ch. masquée

cr	entier	1	écriture réussie
		2	erreur écriture

REMARQUE : Cette fonction doit être suivie d'un SEND ("ENT") pour que la mise à jour soit effective.

8.15. Erreurs

Lorsque l'interpréteur détecte une erreur, il interrompt l'exécution du script et édite un message comprenant le plus souvent le numéro de la ligne erronée et, si possible, un diagnostic.

Les erreurs se répartissent en trois groupes : les erreurs de contexte, les erreurs de syntaxe et les erreurs à l'exécution. Les deux premiers comprennent les contrôles effectués lors de la lecture du code source : un suivi des types des variables, de leur utilisation, des branchements et des étiquettes et le contrôle de la syntaxe. Le troisième groupe correspond aux erreurs survenant lors de l'exécution même du programme.

8.15.1. Erreurs de contexte

La principale caractéristique des erreurs de contexte est de ne pas provoquer l'arrêt de la lecture du programme : lorsqu'une erreur de contexte est détectée, un message d'erreur est édité, et la lecture du programme se poursuit. En revanche, le programme ne sera pas exécuté, il doit auparavant être corrigé.

L'interpréteur détecte les incohérences ou le non-respect des règles concernant les expressions, les affectations, les paramètres. Ces règles ont été indiquées dans la description du langage.

Au niveau des variables, plusieurs contrôles sont effectués. L'utilisation de la valeur d'une variable non initialisée (c'est-à-dire n'ayant subi ni affectation directe, ni utilisation comme paramètre par variable d'une fonction prédéfinie) provoque une erreur. L'utilisation d'une variable non déclarée de même. Si une variable est déclarée mais pas utilisée dans le programme, le fait est signalé, sans qu'il y ait erreur.

Les branchements inconditionnels (GOTO) sont soumis à des règles de contexte strictes. L'utilisation d'étiquettes non déclarées provoque une erreur, ainsi que les branchements vers l'intérieur d'un bloc d'instructions. Si une étiquette est déclarée mais pas utilisée dans le programme, le fait est signalé, sans qu'il y ait erreur.

8.15.2. Erreurs de syntaxe

Lorsque la **syntaxe** n'est pas respectée, l'interpréteur rend immédiatement la main à l'utilisateur et édite un message d'erreur. Le message d'erreur indique le numéro de la ligne où l'erreur est constatée. Ce numéro correspond, non pas à la ligne qu'il faut corriger, mais à celle à partir de laquelle on peut affirmer sans ambiguïté que la syntaxe est incorrecte. Dans l'exemple, l'erreur ne sera détectée qu'à la dernière ligne, alors qu'il manque un END à la ligne 12 :

```
1   PROG NOM_DE_PROG
2   CHAINE SELECTEUR
3   BEGIN
4           SELECTEUR="OUVERT"
5           IF (SELECTEUR=="OUVERT")
6               BEGIN
7                   WRITEM ("LANCEMENT")
8               END
9           ELSE
10          BEGIN
11              WRITEM ("ATTENTE")
12              PAUSE (25)
13  END
```

L'erreur ne peut être détectée plus tôt car l'interpréteur ne peut déterminer à l'avance le nombre de lignes contenues dans le bloc «ELSE».

Les erreurs les plus communément rencontrées sont :

- l'oubli ou l'ajout d'un délimiteur de bloc, BEGIN ou END ;
- l'oubli d'un passage à la ligne obligatoire ;
- l'ajout d'une ligne blanche avant le ELSE ;
- l'ajout ou l'oubli de parenthèses dans les expressions complexes (vérifier qu'elles vont bien par paires) ;
- l'oubli des parenthèses pour une condition ;
- l'oubli du END final.

8.15.3. Erreurs à l'exécution

Lors de l'exécution du programme peuvent apparaître des problèmes non prévisibles à la seule vue du code.

Selon la longueur, la complexité du code et la présence d'algorithmes erronés, les limitations physiques de la machine peuvent entrer en ligne de compte. Ces limitations visent la mémoire et le temps maximal d'exécution des routines d'un utilisateur ; elles sont très dépendantes du site et du système d'exploitation. Leur dépassement (boucle infinie par exemple), provoque sur certains sites une erreur explicite, et sur d'autres un comportement aberrant de la machine.

Les calculs arithmétiques peuvent engendrer des dépassements de la valeur absolue maximale pour un entier (32767). Ceci mène à des résultats erronés mais pas toujours de façon évidente (le calcul de a+b-c, avec a, b et c grands peut faire intervenir une erreur dans le calcul de a+b, alors que le calcul de a+(b-c) peut se dérouler sans problème). L'interpréteur arrête l'exécution dans tout dépassement arithmétique (le message d'erreur n'indique pas le numéro de la ligne fautive).