Connect:Express® UNIX

Mise en Oeuvre des Interfaces d'Exploitation

Version 1.4.5

Sterling Commerce
An IBM Company

Connect:Express UNIX Présentation des Interfaces d'Exploitation
Version 1.4.5
Première Edition .

La présente documentation a pour objet d'aider les utilisateurs autorisés du système Connect:Express (ci-après le « Logiciel de Sterling Commerce »). Le Logiciel de Sterling Commerce, la documentation correspondante ainsi que les informations et le savoir-faire qu'il contient, sont la propriété de Sterling Commerce Inc. et sont confidentiels. Ils constituent des secrets commerciaux de cette dernière, de ses sociétés affiliées ou de ses/leurs concédants (ci-après dénommés collectivement « Sterling Commerce »). Ils ne peuvent pas être utilisés à des fins non autorisées ni divulgués à des tiers sans l'accord écrit préalable de Sterling Commerce. Le Logiciel de Sterling Commerce ainsi que les informations et le savoir-faire qu'il contient ont été fournis conformément à un contrat de licence qui inclut des interdictions et/ou des limitations quant à la copie, la modification et l'utilisation. La reproduction, en tout ou partie, si et lorsqu'elle est autorisée, devra inclure la présente notice d'information et la légende de copyright de Sterling Commerce Inc. Lorsqu'un Logiciel de Sterling Commerce ou un Logiciel Tiers est utilisé, reproduit ou divulgué par ou à une administration des Etats-Unis ou un cocontractant ou sous-traitant d'une telle administration, le Logiciel est assorti de DROITS LIMITES tels que définis au Titre 48 CFR 52.227-19 et est régi par les dispositions suivantes : Titre 48 CFR 2.101, 12.212, 52.227-19, 227-7201 à 227.7202-4, FAR 52.227-14 (g) (2) (6/87) et FAR 52.227-19 (c) (2) et (6/87), et le cas échéant, la licence habituelle de Sterling Commerce, tel que cela est décrit au Titre 48 CFR 227-7202-3 concernant les logiciels commerciaux et la documentation des logiciels commerciaux, y compris le DFAR 252-227-7013 (c) (1), 252.227-7015 (b) et (2), DFAR 252.227-7015 (b) (6/95), DFAR 227.7202-3 (a), selon le cas.

Le Logiciel de Sterling Commerce et la documentation correspondante sont concédés « EN L'ETAT » ou assortis d'une garantie limitée, telle que décrite dans le contrat de licence de Sterling Commerce. A l'exception des garanties limitées accordées, AUCUNE AUTRE GARANTIE EXPRESSE OU IMPLICITE N'EST CONCEDEE, Y COMPRIS LES GARANTIES DE QUALITE MARCHANDE ET DE CONVENANCE A UN USAGE PARTICULIER. La société Sterling Commerce concernée se réserve le droit de revoir cette publication périodiquement et d'effectuer des modifications quant à son contenu, sans obligation d'en informer qui que ce soit, personne physique ou personne morale.

Les références faites dans le présent manuel aux produits, logiciels ou services Sterling Commerce ne signifient pas que Sterling Commerce a l'intention de les commercialiser dans tous les pays dans lesquels elle a des activités.

Imprimé aux Etats-Unis.

Table des matières

Préface

Ce document décrit les interfaces de programmation en langage C et les utilitaires Unix mis à disposition de l'utilisateur du moniteur de transferts de fichiers Connect:Express Unix afin qu'il puisse personnaliser et intégrer les opérations de transferts dans son environnement applicatif.

Ce document s'adresse à un programmeur Unix et suppose connues les fonctionalités de Connect:Express Unix.

Documentation Connect:Express

CONNECT:Express Unix – User Guide
Ce document en Anglais est destiné à un administrateur Unix en charge de l'installation et de la configuration de Connect:Express Unix. Il s'adresse aussi à un utilisateur en charge de la mise en œuvre des transferts de fichiers.

CONNECT:Express Unix – FTP User's Guide
Ce document en Anglais est destiné à un administrateur Unix en charge de l'installation et de la configuration de l'option FTP de Connect:Express Unix. Il s'adresse aussi à un utilisateur en charge de la mise en œuvre des transferts de fichiers FTP.

CONNECT:Express Unix – Utilisation des Pi37 et 99 avec un Partenaire peSIT
Ce document en Français est destiné à un utilisateur de Connect:Expresss Unix en charge de la mise en œuvre des transferts de fichiers et de leur intégration dans l'applicatif. Ce document précise les flux d'information et vous permet de comprendre à quel moment les Pi37 et 99 sont disponibles et modifiables.

CONNECT:Express Unix – Option ETEBAC3
Ce document en Français est destiné à un administrateur Unix en charge de l'installation et de la configuration de l'option ETEBAC3 de Connect:Express Unix. Il est destiné aussi à un utilisateur en charge de la mise en œuvre des transferts de fichiers ETEBAC3.

CONNECT:Express Unix – Mise en œuvre de l'utilitaire de notification
Ce document en Français est destiné à un administrateur Unix en charge de l'installation et de la configuration de l'option Notifications de Connect:Express Unix.

Chapitre 1 - Généralités

Ce chapitre donne un aperçu des interfaces et les utilitaires mis à disposition de l'utilisateur du moniteur de transferts de fichiers Connect:Express Unix . Il définit les différentes méthodes d'implémentation et les fonctions assurées par chacune d'entre elles.

Les chapitres suivants décrivent chaque interface dans le détail et donnent des exemples d'implémentation .

Présentation Générale des interfaces et des outils

L'ensemble des interfaces permet le contrôle en ligne des opérations.

L'interface de programmation peut être mise en œuvre en langage C. Elle ouvre l'accès à l'ensemble des fonctions de paramétrage et de mise en œuvre des transferts de fichiers.

Les utilitaires s'appuient sur cette interface et permettent de définir et de gérer les transferts à partir de shell scripts.

Des exits ou procédures utilisateur peuvent être activés en début et/ou fin de transfert, de façon synchrone pour les uns, asynchrone pour les autres. Une procédure "Généralisée" est dédiée à la gestion des incidents.

Les tables de transcodage externes permettent de supporter des échanges de données avec toutes les plates-formes.

Les commandes d'exploitation permettent d'agir sur l'environnement du moniteur.

Un exit spécifique pour le protocole ETEBAC3 permet de gérer la carte ETEBAC sous n'importe quel format.

Les différents objets sont stockés dans les répertoires du produit :

Répertoire	Objet
/itom	Interface de programmation et utilitaires Exemples
/exit	Exits et procédures utilisateur Procédure généralisée UEXERR Exemples Fichiers en entrée des exits et procédures utilisateur Fichiers en sortie des exits et procédures utilisateur
/config	Tables de transcodage, commandes d'exploitation

L' Interface de programmation

Le mécanisme utilisé repose sur une connexion à la "Message Queue" du moniteur, la dépose d'une demande de service et l'attente de la réponse de Connect:Express.

Une structure, ZREQ_TOM contenue dans le fichier d0b8z20.h, commune à toutes les fonctions, permet ce dialogue, via le module interface L0B8Z20 qui doit être "Link-édité" avec le programme applicatif.

L' API est constituée d'un module interface - L0B8Z20 - et d'une structure de communication - d0b8z20.h

Les services disponibles via cette interface sont les suivants:

Gestion des Répertoires

Les répertoires des Partenaires et des Fichiers symboliques peuvent être mis à jour depuis une application: les fonctions ajout, modification, suppression et visualisation sont disponibles.

Gestion des Transferts

Une application peut déposer une requête de transfert, l'interrompre, la relancer, la purger ou la consulter. Une purge par critères est aussi proposée.

Gestion des Statistiques

Une application peut interroger Connect :Express sur le résultat d'un transfert.

Les utilitaires

Les utilitaires proposés sont des programmes batch qui s'appuient sur l'interface décrite précédemment.

Ces programmes peuvent être appelés depuis un schell. Les paramètres sont passés par mots clés, selon une syntaxe du type suivant:

p1b8preq "/SFN='fichier symbolique'/PRT=1/LNK=T/SPN='Partenaire symbolique' "

Note: Ces programmes, écrits en langage C, sont fournis en exemple d'utilisation de l'interface.

Les services disponibles via ces utilitaires sont les suivants:

Gestion des Répertoires

Les répertoires des Partenaires et des Fichiers symboliques peuvent être modifiés et visualisés depuis un shell. La suppression et la visualisation d'une définition de partenaire ou de fichier est disponible. L'ajout et la modification d'un partenaire et d'un fichier sont fournis en programmes sources.

Les programmes à utiliser, pour la suppression et la visualisation sont respectivement:

Gestion des Répertoires : p1b8ppar_s, p1b8ppar_d, p1b8pfil_s, p1b8pfil_d

Gestion des Transferts

Un shell peut être utilisé pour déposer une requête de transfert, l'interrompre, la purger, ou demander une reprise.

Les programmes à utiliser sont respectivement:

Gestion des Transferts : p1b8preq, p1b8pcan, p1b8ppur, p1b8pret

Gestion des Statistiques

Un shell peut être utilisé pour consulter les résultats statistiques d'un transfert ou d'un ensemble de transferts.

Le programme à utiliser est le suivant:

Gestion des Statistiques : p1b8pren

L'Interface Exit

La définition d'un fichier symbolique permet d'intégrer des exits utilisateur à l'exécution d'un transfert.

Il est possible d'associer à un fichier quatre exits:

Exit de début de réception
Exit de fin de réception
Exit de début d'émission
Exit de fin d'émission

Un exit est un programme exécutable en langage C appelé de façon synchrone. Il fait donc partie intégrante du traitement. Ceci caractérise un exit par rapport à une procédure intégrée telle que la décrit le chapitre suivant.

L'exit reçoit en entrée une structure appelée d1b8ruex.h qui fournit en début de transfert les identifications relatives a ce transfert, et qui est complétée en fin de transfert par les résultats et les codes retours.

L'exit traite une structure de communication - d1b8ruex.h

Les services disponibles via l'interface exit sont les suivants:

Gestion des Noms Physiques de Fichier

En début de transfert, l'exit peut être utilisé pour déterminer un nom physique de fichier. Ce nom est normalement déterminé par Connect:Express à partir de la définition du fichier symbolique, mais il peut être nécessaire, pour des raisons applicatives, de le déterminer au moment du transfert. L'exit reçoit en entrée le nom déterminé par Connect:Express et Connect:Express récupère en retour de l'exit le même champ éventuellement modifié.

Traitements des Champs "Libres" PeSIT

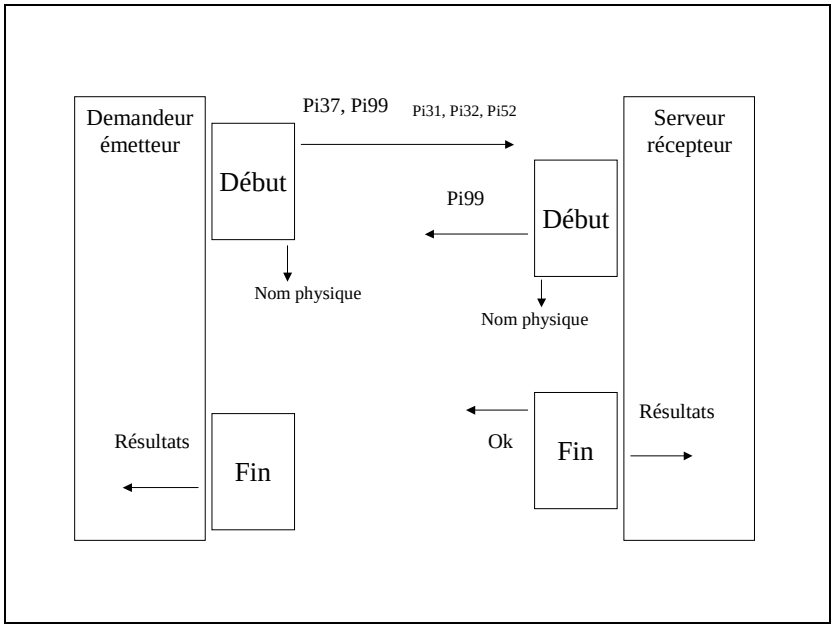
Le mécanisme de la communication entre l'exit et Connect:Express permet de positionner le Pi99 et le Pi37 en début de transfert ou de les traiter en fin de transfert. D'autres paramètres PeSIT peuvent aussi être modifiés par l'exit : Pi3 bis, Pi4 bis, Pi31, Pi32, Pi52, Pi61, Pi62 et le nom physique de fichier.

Suivi des Transferts

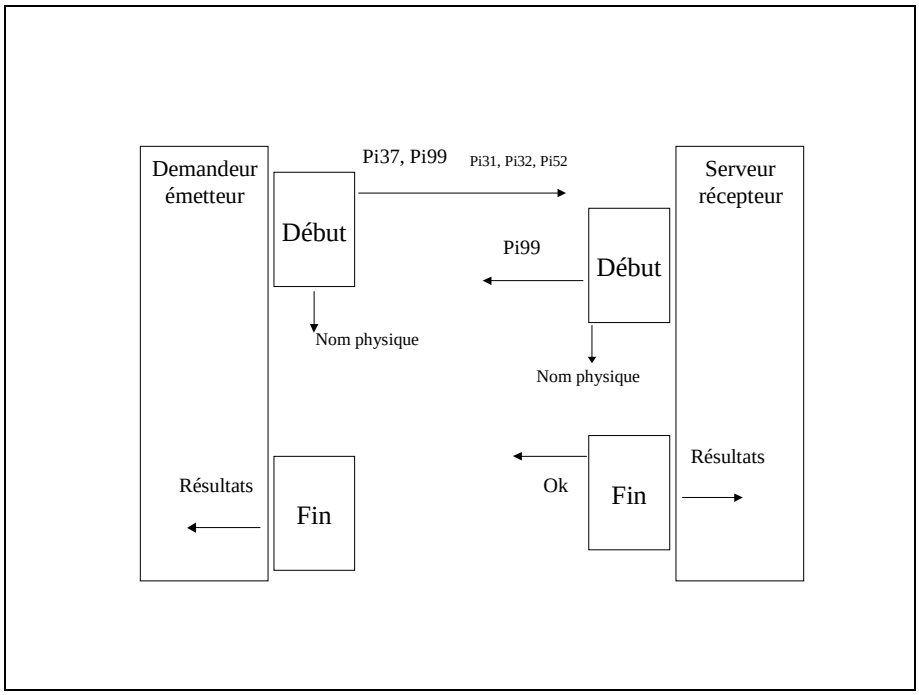
L'exit de fin de transfert permet de contrôler les opérations sur le vif, à partir des codes retour et des résultats statistiques fournis (Nombre d'octets, nombre d'enregistrements, durée, etc …).

Les schémas suivants représentent les étapes du déroulement d'un transfert PeSIT et les points d'ancrage des fonctions d'exit, dans le cas d'une demande d'émission, puis dans le cas d'une demande de réception.

Demande d'émission



Demande de réception



Dans le cas d'un transfert avec le protocole PeSIT, l'exit de début de transfert permet à l'utilisateur de positionner le Pi37, du côté émetteur, ainsi que les pi31, 32 et 52 qui décrivent le fichier émis. Cet exit permet aussi de positionner le Pi99, de format libre dans le cas où le partenaire est de type other. L'exit de début permet enfin, quel que soit le protocole, de déterminer le nom physique du fichier local à transférer. Quel que soit le protocole, l'exit de fin de transfert permet de valider le transfert, du côté récepteur. Cet exit reçoit les résultats aussi bien en cas de succès qu'en cas d'échec.

Les Procédures Utilisateur Intégrées

Contrairement aux exits décrits dans le paragraphe précédent, une procédure intégrée est lancée par le moniteur et s'exécute hors des opérations de transfert. C'est une procédure shell qui permet d'activer un traitement conjoint parallèlement aux opérations de transfert.

La définition d'un fichier symbolique permet d'intégrer des procédures utilisateur à l'exécution d'un transfert.

Il est possible d'associer à un fichier quatre procédures:

Procédure de début de réception
Procédure de fin de réception
Procédure de début d'émission
Procédure de fin d'émission

Une procédure de fin de transfert n'est activée qu'en cas de succès. La procédure généralisée UEXERR fournie permet de gérer les incidents indépendamment du fichier symbolique. Elle est activée en cas d'échec en phase de connexion ou d'interruption de transfert.

Son nom est fixe, l'utilisateur l'implémente comme il le souhaite:

Procédure Généralisée: UEXERR

Les services disponibles via cette interface sont les suivants:

Traitements Associés aux Transferts

Le caractère asynchrone du processus conditionne l'utilisation de telles procédures qui n'ont pas d'influence sur le transfert lui même.

Gestion des Incidents – Procédure Généralisée

Les incidents de transfert peuvent être gérés de façon spécifique pour un fichier, par l'association d'une commande de fin de transfert à ce fichier. Ils peuvent être gérés de façon globale par une procédure unique : la procédure généralisée est activée à chaque incident, dès la phase de connexion. La procédure UEXERR livrée avec le produit n'effectue aucun traitement. C'est à l'utilisateur d'y intégrer les traitements spécifiques. Cette procédure permet un suivi des incidents réseau aussi bien que des incidents de transfert.

Tables de Transcodage

Le transfert de données en milieu hétérogène peut conduire à la nécessité de transcoder des données ASCII en EBCDIC avant émission ou des données EBCDIC en ASCII après réception.

La multiplicité des environnements conduit à utiliser plusieurs tables de transcodage. A partir d'un modèle fourni, l'utilisateur peut constituer une table spécifique et la référence à cette table dans la définition d'un fichier symbolique suffira à déclencher un transcodage conforme à un système de codage spécifique.

Commandes d'exploitation

Arrêt en douceur du moniteur.

Description :

Mise en œuvre d'un arrêt de CONNECT EXPRESS en douceur, permettant l'achèvement des transferts en cours et assurant la mise en attente de tout nouveau transfert.
Cette nouvelle fonction nécessite la création et la gestion de deux nouveaux compteurs pour comptabiliser le nombre de sessions entrante (IN) et sortante (OUT) en cours d'exécution.

Lancement de la demande d'arrêt en douceur du moniteur soit :

par la variable d'environnement $stop_tom_l
par la commande $TOM_DIR /config/stop_tom L

Rechargement des paramètres de configuration par commande au moniteur

Description

Mise à disposition d'une commande opérateur, ch_conf, permettant de modifier en cours de session (sans arrêt-relance du moniteur) certains paramètres de configuration initialisés dans le fichier de configuration sysin, de recharger les fichiers de configuration SYSTCP et SYSX25, ou de charger une nouvelle clé de license.

Lancement de la demande de changement de configuration soit :

par la variable d'environnement $ch_conf
par le lancement de la procédure shell $TOM_DIR/config/ch_conf.sh
par le lancement de la commande $TOM_DIR/config/ch_conf /argument

Procedure ch_conf.sh

```
$ch_conf

************************************************************
*       Modification des parametres de configuration       *
************************************************************
*                                                          *
* Choix 1 --> Activation de la trace                       *
*                                                          *
* Choix 2 --> Arret de la trace                            *
*                                                          *
* Choix 3 --> Lancement de la reorganisation de la base    *
*             au prochain arret du moniteur                *
*                                                          *
* Choix 4 --> Non lancement de la reorganisation de la base *
*             au prochain arret du moniteur                *
*                                                          *
* Choix 5 --> Rechargement du fichier SYSTCP               *
*                                                          *
* Choix 6 --> Rechargement du fichier SYSX25               *
*                                                          *
* Choix 7 --> Rechargement du fichier license.key          *
*                                                          *
* Choix x --> Sortir du menu                               *
*                                                          *
************************************************************

Taper votre choix :
```

Arguments de la commande ch_conf

./ch_conf /STRACE=1 → **Activation de la trace.**

./ch_conf /STRACE=0 → **Arrêt de la trace.**

./ch_conf /RBUILD=1 → **Lancement de la réorganisation de la base au prochain arrêt du moniteur.**

./ch_conf /RBUILD=0 → **Non lancement de la réorganisation de la base au prochain arrêt du moniteur.**

./ch_conf /SYSTCP → **Rechargement du fichier de configuration SYSTCP**

./ch_conf /SYSX25 → **Rechargement du fichier de configuration SYSX25**

./ch_conf /APSKEY → **Rechargement du fichier de configuration license.key**

Exit ETEBAC3

Dans le contexte particulier du protocole de transfert de fichier ETEBAC3, il est possible de supporter des cartes ETEBAC3 de format quelconque.

En mode serveur la carte recue est interprétée par l'exit et traitée sous contrôle applicatif. Un groupe d'informations (nom symbolique de partenaire, mot de passe symbolique, nom symbolique de fichier) est rendu à Connect:Express en sortie de l'exit pour traitement standard par le moniteur.

En mode demandeur il est possible de construire une carte ETEBAC3 spécifique associée à un couple (Partenaire symbolique, Fichier symbolique).

Se reporter à la documentation "*Connect:Express for UNIX – Option ETEBAC3*".pour plus d'information

Chapitre 2 - API

Ce chapitre décrit les principes généraux de mise en œuvre de l'interface de programmation et l'illustre par un exemple de requête de transfert.

Présentation de l'Interface de Programmation

Le moniteur peut être administré à l'aide d'une interface de programmation écrite en langage C.

Les différents appels de l'interface de programmation se font au travers d'une fonction nommée L0B8Z20, en lui passant en paramètre une structure ZREQ_TOM dont certains champs ont été préalablement renseignés. Au retour de la fonction, la structure ZREQ_TOM contient les codes retour de l'appel ainsi que des valeurs de retour le cas échéant.

Le tableau ci dessous indique les composants fournis dans les répertoires itom et itom/SAMPLES :

Module	Description
libitom.a	Bibliothèque contenant l'objet LOB8Z20.o
d0b8z20.h Cette	Structure de communication entre un programme applicatif et l'interface de Connect:Express. structure contient les descriptions des fichiers de Connect :Express (RPAR,RFIC et RENC par exemple).
SAMPLES	Répertoire de fichiers sources
p1b8pcan.c	Fichier source c : exemple d'interruption de transfert
p1b8pfil_c.c	Fichier source c : exemple de création d'un fichier
p1b8pfil_d.c	Fichier source c : exemple de visualisation d'un fichier
p1b8pfil_m.c	Fichier source c : exemple de mise à jour d'un partenaire
p1b8pfil_s.c	Fichier source c : exemple de suppression d'un fichier
p1b8ppar_c.c	Fichier source c : exemple de création d'un partenaire
p1b8ppar_d.c	Fichier source c : exemple de visualisation d'un partenaire
p1b8ppar_m.c	Fichier source c : exemple de mise à jour d'un partenaire
etc …	

Un programme C écrit par l'utilisateur peut demander au moniteur d'effectuer différents type d'opérations tels que:

Gestion des transferts (fichier RENC)

Requête de transfert
Interruption d'un transfert donné
Purge d'un ou de plusieurs transferts suivant certains critères
Redémarrage d'un transfert donné
Affichage des données concernant un transfert

Gestion des partenaires (fichier RPAR)

Des programmes sources sont fournis en exemples : p1b8ppar_c et p1b8ppar_m.

Affichage des caractéristiques d'un partenaire symbolique
Création, modification et suppression d'un partenaire symbolique

Gestion des fichiers (fichier RFIC)

Des programmes sources sont fournis en exemples : p1b8pfil_c et p1b8pfil_m.

Affichage des caractéristiques d'un fichier symbolique
Création, modification et suppression d'un fichier symbolique

Gestion des statistiques (fichier RENC)

Affichage d'une requête de transfert

Mise en Œuvre de l'Interface

La définition de la structure ZREQ_TOM utilisée par l'interface de programmation est décrite dans le fichier d0b8z20.h qui devra être inclus dans le source du programme C utilisateur. La compilation du programme utilisateur devra inclure l'option suivante :

-L $TOM_DIR/itom

Cette option pointe sur la variable d'environnement correspondant au répertoire racine du moniteur (par exemple /home/tom1).

La fonction L0B8Z20 appelée par le programme utilisateur est incluse dans la bibliothèque d'objets « libitom.a » située dans le répertoire désigné par la variable d'environnement $TOM_DIR/itom. Il doit être link-édité avec le programme appelant.

Un exemple de script de compilation et un exemple de fichier de MAKE de programme utilisateur sont fournis en Annexe 3.

Description de la Structure ZREQ_TOM

La structure ZREQ_TOM se compose d'une en-tête commune aux différents types d'utilisation de l'API, puis de l'union de structures plus spécifiques dépendant du type de l'appel à l'API.

structure	st_sci	Demande de transfert au moniteur
structure	s_renc	Actions sur la base des encours
structure	partenaire	Actions sur la base des partenaires
structure	fichier	Actions sur la base des profils de fichier

(Voir l' annexe 1 pour la définition détaillée donnée par le fichier d0b8z20.h)

En-tête

```
struct ZREQ_TOM {
     char zreq_tom_name[4]; /* Nom du moniteur          */
     char zreq_tom_func[1]; /* Type de fonction         */
     char zreq_tom_tabn[1]; /* Type d'appel à l'API     */
     char zreq_tom_reqn[8]; /* Numéro de requête        */
     char zreq_tom_rtcf[1]; /* Code retour du moniteur  */
     char zreq_tom_rscf[3]; /* Reason return code       */
     union uni_sci uni;
};
```

Structure d'appel

```
union  uni_sci {
     struct       st_sci zreq_tom_sci;
     struct       s_renc zreq_tom_renc;
     struct       partenaire zreq_tom_part;
     struct       fichier zreq_tom_fic;
};
```

L'en tête indique par un premier code le type de fonction (Transfert, Modification, Suppression ...), et par un sous code le fichier accédé (Requêtes, Fichiers, Partenaires). Le tableau suivant fait la synthèse des services disponibles :

Type de fonction	Fichier accédé	Service
T = Requête de Transfert	R = Fichier des Requêtes : RENC	**TR**
I = Interruption d'un transfert		**IR**
P = Purge d'un transfert		**PR**
R = Reprise d'un transfert		**RR**
interrompu		**DR**
D = Display des informations		
C = Création	P = Fichier des Partenaires :	**CP**
M = Modification	RPAR	**MP**
S = suppression		**SP**
D = Display des informations		**DP**
C = Création	F = Fichier des Fichiers : RFIC	**CF**
M = Modification		**MF**
S = Suppression		**SF**
D = Display des informations		**DF**

Exemple Type de Programme

Un programme applicatif simple pourra être écrit selon la logique suivante :

```
#include <d0b8z20.h>
ZREQ_TOM *zreq;
..................

zreq = malloc(sizeof(ZREQ_TOM));

/* Chargement des valeurs d'appels adéquates dans la structure zreq en
fonction de l'action voulue */
..................

ret = L0B8Z20(zreq).

/* Test des codes retournés dans la structure zreq */
..................
free(zreq);
..................
exit(0);
```

L'ensemble des fonctions proposées par l'interface permettent le contrôle en ligne ou le suivi statistique.

Fonctionnement de la Communication avec le Moniteur

La communication avec le moniteur se fait à l'aide de messages queues (IPC système V). Le programme utilisateur (à l'intérieur de la fonction L0B8Z20) se crée une message queue. Puis il dépose sa requête dans la message queue du moniteur (dont l'identifiant est TOM_DIR). Cette requête contient entre autre l'identifiant de la message queue du programme utilisateur sur laquelle le moniteur déposera les résultats de la requête après traitement. Avant de rendre la main, la fonction L0B8Z20 supprime la message queue correspondant à l'appel.

Cette gestion des message queues est transparente pour l'utilisateur qui n'a pas à s'en soucier. Néanmoins, ceci montre qu'un programme utilisant l'interface ne pourra fonctionner que si le moniteur est actif.

Gestion des Transferts

La gestion des transferts comprend le dépôt d'une requête, l'interruption et la purge d'une requête, le reprise d'un transfert interrompu et l'affichage des informations relatives à un transfert.

Le dépôt d'une requête de transfert se traduit par l'attribution d'un numéro de requête obtenu en retour de l'interface. Ce numéro de requête est utilisé ultérieurement pour les autres fonctions appliquées à cette requête.

Exemple: Requête de Transfert

int L0B8Z20(struct ZREQ_TOM *);

Description

Cette fonction permet de déposer dans la message queue du moniteur une demande de transfert et d'en préciser les différentes caractéristiques : sens, nom symbolique de fichier, nom partenaire, etc…

Paramètres d'Appel

L0B8Z20 est appelé avec, en paramètre, une structure ZREQ_TOM dont certains champs sont renseignés.

La taille en nombre caractères de chaque champ utilisé est indiquée entre crochets. Par exemple zreq_tom_name[4] a pour longueur 4 caractères.
Les paramètres qui ne sont pas mentionnés « obligatoires » sont facultatifs dans la mesure où des valeurs par défaut correspondantes existent. Ces valeurs peuvent provenir notamment de la définition du nom symbolique de FICHIER).
Si ces valeurs ne sont pas renseignées dans l'appel, elles doivent être initialisées, sauf indication contraire, à « » (espace).
Les paramètres utilisés sont identiques à ceux utilisés par l'utilitaire STERM pour effectuer une demande de transfert. On pourra éventuellement consulter la documentation de STERM pour avoir de plus amples renseignements sur la signification de ceux-ci.

Les tableaux ci-dessous décrivent les paramètres d'appel, l'en tête puis la partie spécifique liée au service demandé, combinaison du code et du sous code fonction.

En-tête:

Champ	Description	Remarque
zreq_tom_name[4]	Nom du moniteur . (En général « TOM1 »)	Obligatoire
zreq_tom_func[1]	Code fonction de l'API « T » Service de Transfert	Obligatoire
zreq_tom_tabn[n]	Code sous-fonction « R » Fichier des Requêtes (RENC)	Obligatoire
zreq_tom_reqn[8]:	Numéro de requête. Doit être renseigné à zéro binaire (x'0') à l'appel	Réponse
zreq_tom_rtcf[1]:	Code retour du moniteur. Doit être renseigné à zéro binaire (x'0') à l'appel	Réponse
zreq_tom_rscf[3]	Code raison complémentaire. Doit être renseigné à zéro binaire (x'0') à l'appel	Réponse

Partie spécifique:

Champ	Description	Remarque
uni.zreq_tom_sci.dire[1]	Sens du transfert. Valeurs « T » (Transmission) ou « R » (Réception).	Si non renseigné (espaces), la valeur prise en compte sera celle du profil fichier.
uni.zreq_tom_sci.file[8]	Nom du profil fichier (Nom symbolique fichier). 1 à 8 caractères alphanumériques. Complétés par des espaces	Obligatoire.
uni.zreq_tom_sci.part[8]	Nom symbolique de partenaire. 1 à 8 caractères alphanumériques. Complétés par des espaces.	Si non renseigné (espaces), la valeur prise en compte sera celle du profil fichier.
uni.zreq_tom_sci.dsnam[44]	Nom physique de fichier. 1 à 44 caractères alphanumériques. Complétés par des espaces.	Si non renseigné (espaces), la valeur prise en compte sera celle du profil fichier.
uni.zreq_tom_sci.prty[1]	Priorité de transfert. Caractères « 0 » (x'30'), « 1 » (X'31') ou « 2 » (X'32').	Si non renseigné (espaces), la valeur prise en compte sera celle du profil fichier.
uni.zreq_tom_sci.dat[8]	Date demandée pour le début du transfert.	Si date et heure sont toutes deux non renseignées

Champ	Description	Remarque
	Au format AAAAMMJJ.	(espaces), la demande est prise en compte immédiatement.
uni.zreq_tom_sci.hour[6]	Heure demandée pour le début du transfert. Au format HHMMJJ.	Si date et heure sont toutes deux non renseignées (espaces), la demande est prise en compte immédiatement.
uni.zreq_tom_sci.lnk[1]	Type d'interface réseau utilisée. Valeurs « T » (TCP/IP), « X » (X25) ou « P » (PAD).	Si non renseigné (espace), la valeur prise en compte sera celle du partenaire
uni.zreq_tom_sci.udf[44]	Données utilisateur.	
uni.zreq_tom_sci.typ[1]	Type de demande. Valeurs : « N » (Normal) « I » Inquiry : Permet de recevoir un fichier mis à dispostion par un partenaire distant à l'aide d'une demande de type « H ». (En mode demandeur récepteur uniquement). « H » Hold : Mise à disposition de fichier. La demande sera selectionnée par le partenaire distant à l'aide d'une demande de type « I ». (En mode serveur émetteur uniquement).	
uni.zreq_tom_sci.sta[1]	Ignoré	Etat du transfert
uni.zreq_tom_sci.dpcsid[8]	Alias DPCSID. Nom du moniteur local tel qu'il se presentera au moniteur distant.	Si non renseigné (espaces), la valeur prise en compte sera celle du partenaire. Si la valeur n'est pas renseignée pour le partenaire, la valeur prise en compte sera celle du DPCSID du fichier sysin.
uni.zreq_tom_sci.dpcpsw[	Alias DPCPSW. Mot de passe du moniteur local tel qu'il se	Si non renseigné (espaces), la valeur

Champ	Description	Remarque
8]	présentera au moniteur distant.	prise en compte sera celle du partenaire. Si la valeur n'est pas renseignée pour le partenaire, la valeur prise en compte sera celle du DPCSID du fichier sysin.
uni.zreq_tom_sci.format[2] :	Format des enregistrements du fichier. « TF » Format texte. Enregistrements de longueur fixe. « TV » Format texte. Enregistrements de longueur variable. « BF » Format binaire. Enregistrement de longueur fixe. « BU » Format binaire indéfini.	Si non renseigné (espaces), la valeur prise en compte sera celle du profil fichier
uni.zreq_tom_sci.lrecl[5]	Longueur d'enregistrement. 5 caractères numériques.	Si non renseigné (espaces), la valeur prise en compte est celle du profil fichier.
uni.zreq_tom_sci.api[88]	Champ api. En Etebac3, ce champ correspond à la carte Etebac3.	
uni.zreq_tom_sci.tsm[3]	Type, structure, et mode pour transferts en FTP. Type peut prendre les valeurs : « A » (Ascii), « E » Ebcdic, « B » Binary, « * » Inchangé Structure peut prendre les valeurs : « F » File, « R » Record, « * » Inchangé Mode peut prendre les valeurs : « B » Block, « S » Stream, « * » Inchangé	Si non renseigné (espaces), la valeur prise en compte est celle du profil fichier.
uni.zreq_tom_sci.stou[1]	Indicateur store unique pour transferts FTP. Valeurs possibles : « Y » Oui, « N » Non	Si non renseignée (espace), la valeur prise en compte est celle du profil fichier.
uni.zreq_tom_sci.fa[1]	Indicateur pour la prise en compte en réception du fichier par le File Agent de CONNECT :Enterprise.. Valeurs possibles : « Y » Oui, « N » Non	Si non renseignée (espace), la valeur prise en compte est celle du profil fichier.
uni.zreq_tom_sci.label[80]	Pi37 PeSIT	En émission seulement.

Champ	Description	Remarque
uni.zreq_tom_sci.s_pi99_254[254]	Pi99 PeSIT	Avec un partenaire de type other.
uni.zreq_tom_sci.user_org[8]	Nom origine du transfert	Pi3 bis
uni.zreq_tom_sci.user_dst[8]	Nom destinataire du transfert	Pi4 bis
uni.zreq_tom_sci.user_snd[24]	Emetteur du transfert	Pi61
uni.zreq_tom_sci.user_rcv[24]	Recepteur du transfert	Pi62
uni.zreq_tom_sci.quant_aa[2]	Année pour quantième	
uni.zreq_tom_sci.quant[3]	Quantième de l'année	
uni.zreq_tom_sci.notif[1]	Notification	

Valeurs de Retour

La valeur de retour de l'appel à L0B8Z20 peut être testée de la façon suivante :

status = L0B8Z20(param);

Status = 0 : Pas d'erreur
Status = 2 : Erreur

La structure ZREQ_TOM contient elle même le résultat de la demande.

Champ	Description
zreq_tom_reqn[8]:	contient le numéro de requête attribué par le moniteur au transfert .
zreq_tom_rtcf[1]:	si zreq_tom_rtcf[1] = 0 Pas d'erreur si zreq_tom_rtcf[1] <> 0 Erreur.
zreq_tom_rscf[3]	zreq_tom_rscf[3] contient le TRC (Code d'erreur retourné à l'API par le moniteur).

Exemple

```
#include "d0b8z20.h"
------
struct ZREQ_TOM *param;
#define SIZE_ZREQ
int L0B8Z20(struct ZREQ_TOM *);
int status  = 0 ;
void main(int argc,char *argv[])
{

/* Structure de communication */
param = (struct ZREQ_TOM *)malloc(sizeof(struct ZREQ_TOM));
if (param==(struct ZREQ_TOM *)0) exit(1) ;
memset((char *)param, ' ', sizeof(struct ZREQ_TOM));

/* Initialisation de la structure avec les paramètres de la demande */

/* Header */
param->zreq_tom_func[0] = 'T';
param->zreq_tom_tabn[0] = 'R';
memset(param->zreq_tom_reqn, 0x00, 8);
memset(param->zreq_tom_rtcf, 0x00, 4);

/* Partie spécifique */
param->uni.zreq_tom_sci.dire[0] = 'T' ;
memcpy(param->uni.zreq_tom_sci.file, "FILE01",6);
memcpy(param->uni.zreq_tom_sci.part, "PART01",6);
memcpy(param->uni.zreq_tom_sci.dsnam, "/home/tom1/out/f01.txt",22);
memcpy(param->uni.zreq_tom_sci.lnk[0] = 'T' ;

status = L0B8Z20(param);
if (status != 0) {
  /* Request not OK */
  free(param) ;
  fprintf(stderr, "%.4s\n", param->zreq_tom_rtcf);
  fflush(stderr);
  exit(1);
}

printf("%.8s\n",param->zreq_tom_reqn);
free(param) ;
exit(0);
}
```

Chapitre 3 - Utilitaires

Ce chapitre décrit les principes généraux de mise en œuvre des utilitaires et donne des exemples d'appels.

Présentation des Utilitaires

Les utilitaires proposés sont des programmes batch qui s'appuient sur l'interface de programmation décrite au chapitre 1. Ces programmes sont livrés dans le répertoire ITOM.

Liste des Utilitaires

Le tableau ci dessous donne la liste des utilitaires fournis dans le répertoire /itom, leur description et le fichier de Connect:Express concerné : RENC est le fichier des requêtes, RPAR le fichier des partenaires, RFIC le fichiers des fichiers symboliques.

Module Connect:Express	**Description**	**Fichier**
p1b8pcan	Interrompt une requête de transfert.	RENC
p1b8pfil_d	Affiche la définition d'un fichier symbolique.	RFIC
p1b8pfil_dfl	Affiche le format et la longueur d'un fichier symbolique.	RFIC
p1b8pfil_s	Supprime un fichier symbolique.	RFIC
p1b8ppar_d	Affiche la définition d'un partenaire du fichier des partenaires .	RPAR
p1b8ppar_s	Supprime un partenaire du fichier des partenaires .	RPAR
p1b8ppur	Purge une requête de transfert.	RENC
p1b8pren	Demande l'affichage des statistiques d'une requête de transfert.	RENC
p1b8preq	Soumet une requête de transfert.	RENC
p1b8pret	Demande la reprise d' une requête de transfert.	RENC

Paramètres d'Appel

Les mots clés utilisés par les utilitaires sont regroupés dans le tableau suivant. La dernière colonne indique quel fichier de Connect :Express est concerné: RENC est le fichier des requêtes, RPAR le fichier des partenaires, RFIC le fichiers des fichiers symboliques.

Mot Clé	Valeur	Description	Utilisation
/REQ	NOMBRE DE 8 CHIFFRES	Numéro de requête. Exemple : 10400023	RENC
/DAT	YYYYMMDDHHMMSS	Date et Heure données en totalité ou partiellement. Exemple : 20030801, 2003080112	RENC
/SFN	8 caractères alphanumériques	Nom symbolique de fichier. Exemple : RAPPORT	RENC RFIC
/SPN	8 caractères alphanumériques	Nom symbolique de partenaire. Exemple : BORDEAUX	RENC RPAR
/DIR	T , R ,*	Sens de transfert : Transmission, Réception, Les deux sens	RENC
/PRT	0, 1, 2	Priorité	RENC
/LNK	T, X, M	Type de lien réseau X25, TCP/IP ou Mixte	RENC
/DSN	1 à 44 caractères alphanumériques	Nom physique de fichier Unix local	RENC
/UDF	1 à 44 caractères alphanumériques	Données utilisateur. Par exemple, Nom physique de fichier chez le serveur distant	RENC
/MNM	4 caractères	Nom du moniteur. Exemple : TOM1	RENC
/SID	1 à 8 caractères alphanumériques	Nom symbolique local (Alias)	RENC
/PSW	1 à 8 caractères alphanumériques	Mot de passe local (Alias)	RENC
/RFM	TF, BF, TV, BU, T*, B*	Format d'enregistrement : Text, Binaire, Fixe, Variable, Undefined …	RENC
/RLG	1 à 5 caractères numériques	Taille d'enregistrement	RENC
/API	1 à 88 caractères alphanumériques	Carte ETEBAC3	RENC
/TYP	N, H, I	Type de requête : Normale, Hold, Inquiry	RENC
/TSM	A/E/B, F/R, B/S	FTP. Type de données : Structure de fichier , Mode de transfert	RENC
/STO	Y/N	FTP. Option Store Unique : oui/non	RENC
/LAB	1 à 80 caractères alphanumériques	PeSIT : Pi37 , label	RENC
/P99	1 à 254 caractères alphanumériques	PeSIT : Pi99 pour partenaire de type other uniquement	RENC
/ORG	1 à 8 caractères alphanumériques	Nom origine du transfert	RENC
/DST	1 à 8 caractères alphanumériques	Nom destinataire du transfert	RENC
/P61	1 à 24 caractères alphanumériques	Emetteur du transfert	RENC
/P62	1 à 24 caractères alphanumériques	Récepteur du transfert	RENC
/QQQ	Nombre de 3 Chiffres	Quantième de l'année	RENC
/NTF	0,1,2,3,4,5,6,7	Type de Notification	RENC

Valeurs de Retour

Tous les utilitaires rendent les mêmes codes retour. Le code est rendu dans la variable $? du shell.

Code rerour	Description
0	Fontion exécutée avec succès
1	Nombre d'arguments incorrect.
2	Erreur détectée par l'utilitaire: voir le code complémentaire (Tableau suivant).
3	Erreur détectée par Connect:Express. Le Guide Utilisateur donne la liste des codes dans l' *Appendix B Return Codes.*

Le tableau suivant fournit les codes complémentaires associés au code retour 2: ces codes sont de la forme XYYZ, avec X = numéro d'argument de l'appel, YY = champ en erreur (SFN, SPN ….), Z = type d'erreur (Paramètre invalide, paramètre en double …) .

```
/* Internal Error Return Code */
#define ERROR_BAD_FUNC    2900
#define ERROR_CRE_QUEUE   2901
#define ERROR_PB_SEND     2902
#define ERROR_PB_RECV     2903
#define ERROR_TIME_OUT    2904
#define ERROR_NOTOM       2912
#define ERROR_OTHER       2999

/* External Error Status (4 digits) : XYYZ */
/* X : argument number (1,2,3) */
/* YY : Field which contains error */
/* Z : Error type */

/* YY */
#define Y_OTH 0    /* Other               */
#define Y_PRT 1    /* Priority            */
#define Y_DIR 2    /* Direction           */
#define Y_LNK 3    /* Link                */
#define Y_SPN 4    /* Partner             */
#define Y_SFN 5    /* File                */
#define Y_DSN 6    /* Physical Name       */
#define Y_UDF 7    /* User Data Field     */
#define Y_DAT 8    /* Date                */
#define Y_MNM 9    /* Monitor             */
#define Y_REQ 10   /* Request Number      */
#define Y_SID 11   /* Alias Name          */
#define Y_PSW 12   /* Alias Password      */
#define Y_RFM 13   /* Record Format       */
#define Y_RLG 14   /* Record Length       */
#define Y_API 15   /* Api                 */
#define Y_STA 16   /* State               */
#define Y_TYP 17   /* Request Type        */
#define Y_TSM 18   /* Type/Struct/Mode FTP */
#define Y_STO 19   /* Store/Unique FTP    */
#define Y_FAG 20   /* File agent flag Y/N  */
#define Y_LAB 21   /* Label               */
#define Y_P99 22   /* PI99 254 char       */
#define Y_ORG 23   /* User Origin         */
#define Y_DST 24   /* User Destination    */
#define Y_P61 25   /* Pi61                */
#define Y_P62 26   /* Pi62                */
#define Y_QQQ 27   /* Julian Date         */
#define Y_NTF 28   /* Notification        */


/* Z */
#define Z_INV_FIELD 1 /* Invalid Field           */
#define Z_DUP_FIELD 2 /* Duplicate Field         */
#define Z_LG_FIELD  3 /* Invalid Field Length    */
#define Z_MIS_FIELD 4 /* Missing Compulsory Field */
```

Exemple Type de Mise en Oeuvre

Une procédure simple pourra être écrite selon la logique suivante :

```
#
#   Sample to Request a Transfer to Monitor.
#
$TOM_DIR/itom/p1b8preq "/SFN=FILE/SPN=PART1/DIR=T" > reqnumb.txt
code=$?
echo Return Code $code
cat reqnumb.txt
```

Dans cet exemple, le numéro de requête est récupéré dans le fichier reqnumb.txt pour être exploité ultérieurement. Le code retour est récupéré dans la variable $?. Si le transfert a été accepté par Connect:Express, le fichier reqnumb.txt contiendra une valeur.

Exemples

Quelques exemples courants sont donnés dans ce paragraphe. Les traitements sur un partenaire ou sur un fichier sont similaires.

Requête de Transfert

Depuis une procédure shell, l'utilisateur peut déposer une requête de transfert, par exemple transférer le fichier FILE, en priorité 1, sur lien TCP/IP avec le partenaire PART . Le nom de fichier à transférer est TOM.tmp et on envoie la chaîne de caractères 'informations utilisateur' dans le Pi99:

```
$TOM_DIR/itom/p1b8preq "/SFN=FILE/PRT=1/LNK=T/SPN=PART" "/DSN=/tmp/TOM.tmp"
                       "/P99=informations utilisateur "
```

Les paramètres de la requête sont constitués de un à quatre arguments. Le premier argument représente la définition du transfert. Les arguments 2, 3 et 4 sont pris parmi 4 autres possibilités : Nom physique de fichier, label, champ utilisateur Pi99 (/UDF si type Tom ou /P99 si type Other) et champ 'API'.

Argument	Champ	Longueur	Commentaire
Transfer Definition (Arg 1)	Nom symbolique de fichier (/SFN=...)	1 à 8 caractères alphanum.	Obligatoire
	Nom symbolique de partenaire (/SPN=...)	1 à 8 caractères alphanum.	Défaut dans la définition RFIC
	Priorité (/PRT=...)	0, 1, or 2	Défaut dans la définition RFIC
	Type de lien (/LNK=...)	T, P or X	Défaut dans la définition RPAR
	Date et Heure (/DAT=...)	yyyymmddhhmmss	Défaut courant
	Direction (/DIR=...)	T or R	Défaut dans la définition RFIC
	Type de requête (/TYP=...)	N, I or H	Défaut N
	Dpcsid Alias (/SID=...)	1 à 8 caractères alphanum.	Défaut dans la définition RPAR
	Dpcpsw Alias (/PSW=...)	1 à 8 caractères alphanum.	Défaut dans la définition RPAR
	Origine (/ORG=…)	1 à 8 caractères alphanum	Facultatif
	Destination (/DST=…)	1 à 8 caractères alphanum	Facultatif
	Emetteur (/P61=…)	1 à 24 caractères alphanum	Facultatif
	Recepteur (/P62=…)	1 à 24 caractères alphanum	Facultatif
	Format d'article (/RFM=...)	2 caractères alphabétiques (TV, TF, BU, BF)	Défaut dans la définition RFIC
	Taille d'article (/RLG=...)	5 caractères numériques	Défaut dans la définition RFIC
	Format FTP (/TSM=...)	3 caractères alphabétiques A,E,B,* F,S,* B,R,*	Défaut dans la définition RFIC
	Store Unique FTP (/STO=...)	Y,N	Défaut dans la définition RFIC
	Notification (/NTF=…)	0,1,2,3,4,5,6,7	Défaut dans la définition RFIC

Physical Name (Arg 2,3 or 4)	Nom physique de fichier (/DSN=...)	1 à 44 caract. alphanum.	Défaut dans la définition RFIC
User Data Definition (Arg 2,3 or 4)	Données utilisateur (/UDF=...)	1 à 44 caract. alphanum.	Facultatif
Label Definition (Arg 2,3 or 4)	Label (/LAB=...)	1 à 80 caract. alphanum.	Facultatif
P99 Field (Arg 2,3 or 4)	Pi99 (/P99=...)	1 à 254 caract. alphanum.	Facultatif Champ libre PeSIT
Note : *valable uniquement avec des partenaires de type Other*			
API Field (Arg 2,3 or 4)	Champ API (/API=...)	1 à 88 caract. alphanum.	Facultatif Carte Etebac 3

Interruption d'une Requête

Depuis une procédure shell, l'utilisateur peut interrompre un transfert, en passant le numéro de requête de transfert en argument à l'utilitaire P1B8PCAN, selon l'exemple ci dessous:

```
$TOM_DIR/itom/p1b8pcan /REQ=10400065
```

Purge d'une ou Plusieurs Requêtes

Depuis une procédure shell, l'utilisateur peut purger une requête à partir de son numéro, plusieurs requêtes en sur critères ou toutes les requêtes. L'utilitaire P1B8PPUR traite plusieurs arguments:

Argument	Description
➢ *Requête unique:*	
/REQ=QQQNNNNN	Numéro de la requête à purger
➢ *Plusieurs requêtes d'après le filtre :*	
/DAT=YYYYMMDDHHMMSS	Les requêtes antérieures à cette date et heures
/QQQ=AAQQQ	Les requêtes antérieures à cette date
/DIR= (R,T ou *)	Les requêtes dans la direction précisée ou dans les 2 sens
/SFN=	Les requêtes pour un nom symbolique de fichier
/SPN=	Les requêtes pour un nom symbolique de partenaire
/STA= (A,C,D,E,H,J,K ou O)	Les requêtes dans un état spécifique
➢ `Toutes les requêtes :`	`Aucun argument`

Plusieurs exemples de purge de requêtes sont donnés ci après :

Commande	Description
$TOM_DIR/itom/p1b8ppur /REQ=10400065	Purger la requête numéro 10400065
$TOM_DIR/itom/p1b8ppur /DAT=20030801	Purger toutes les requêtes antérieures au premier Août 2003
$TOM_DIR/itom/p1b8ppur /QQQ=05103	Purger toutes les requêtes antérieures au 13 Avril 2005
$TOM_DIR/itom/p1b8ppur /DAT=20030801 /SFN=FIC	Purger toutes les requêtes pour le fichier symbolique FIC et antérieures au premier Août 2003
$TOM_DIR/itom/p1b8ppur /STA=E	Purger toutes les requêtes dont le transfert s'est déroulé sans problème (état E)
$TOM_DIR/itom/p1b8ppur	Vider le fichier RENC des requêtes de transfert

Suppression d’un Partenaire

Depuis une procédure shell, l’utilisateur peut supprimer un partenaire, par son nom symbolique:

```
$TOM_DIR/itom/p1b8ppar_s /SPN=PARTNER
```

Chapitre 4 - Exits et Procédures

Ce chapitre décrit les principes généraux de mise en œuvre des exits et des commandes utilisateurs en début et fin de transfert.

Exits Utilisateurs

Connect:Express peut déclencher des exits utilisateurs en début ou fin de transfert. La définition d'un fichier symbolique permet de préciser, dans le deuxième écran, quel exit déclencher en début et fin de transfert, en émission et en réception. L'écran ci dessous en donne l'exemple :

```
C:E/UNIX 145-1  -------------------- FILES DIRECTORY ----------------------- tom1
OPTION ===>

SYMBOLIC NAME        :             FICHIER     DEFINITION : D DIRECTION : R

TRANSMISSION :
START EXIT ............ : EXTDEBT.....
START COMMAND ......... : .............
END EXIT ............... : EXTENDT.....
END COMMAND ........... : .............

RECEPTION :
START EXIT ............ : EXTDEBR.....
START COMMAND ......... : .............
END EXIT ............... : EXTENDT.....
END COMMAND ........... : .............

DO YOU WANT TO GO ON ?
OPTION : VIEW                          UPD : 20040601112010 C:E 141-3
-ENTER- NEXT FIELD                -F3- CANCEL                -F8- COMPLETION
```

Le tableau suivant indique les composants fournis dans le répertoire /exit :

Module	Description
d1b8uex.h	Structure de communication entre un exit utilisateur et Connect:Express.
chpi37.c	Fichier source : exemple de chargement du Pi37
user.c	Fichier source : affichage des paramètres de transfert

L'exit est lancé par Connect:Express avec une structure de paramètres appelée d1b8ruex.h, écrite dans un fichier temporaire dont le nom est passé en paramètre unique à l'exit. La structure d1b8ruex.h se trouve dans le répertoire /exit. Elle est donnée en annexe 2.

Les exits utilisateurs doivent être copiés dans le répertoire /exit . Si le paramètre STRACE est positionné à 1 dans le fichier SYSIN de Connect :Express, un fichier de trace est créé dans le répertoire /exit. Son nom est de la forme E*x*_QQQnnnnn. La variable *x* peut prendre les valeurs 'I' pour un appel en début de transfert, 'E' pour un appel en fin de transfert sans erreur, 'F' pour un appel en fin de transfert interrompu.
L'exit utilisateur peut modifier certaines valeurs, lorsqu'il est appelé en début de transfert : le nom physique de ficher à transmettre, les champs PeSIT Pi31, Pi32, Pi37, Pi52, Pi61, Pi62, Pi99, Pi3 bis et Pi4 bis.

Connect :Express attend le résultat de l'exit avant de continuer le traitement. En début de transfert il récupère les nouvelles valeurs (Pi31, Pi32, Pi37, Pi52, Pi61, Pi62, Pi99, Pi3 bis et Pi4 bis) avant de les exploiter et de les envoyer au partenaire distant. En fin de transfert l'exit peut provoquer le rejet du transfert.

Exemple Type de Traitement par Exit

Un exit simple pourra être écrit selon la logique suivante : lecture du fichier des paramètres, modification du champ désiré puis ré-écriture du fichier des paramètres.

```
/* ************************************
   * Exemple d'exit de chargement du PI37 *
   ************************************ */
#include <stdio.h>
#include <errno.h>
#include "d1b8ruex.h"

#define SIZE_D1B8 sizeof(struct d1b8ruex)

struct d1b8ruex d1b8;

FILE *param;        /* File Pointer to parameters file */
int bytes;          /* To check File size              */
int i;

main(int argc, char *argv[])
{
param = fopen(argv[1], "r+");
if (param == NULL) {
  perror("");
  printf("Error Opening %s File.\n", argv[1]);
  exit(2);
}

bytes = fread((char *)&d1b8,1,SIZE_D1B8,param);
if (bytes!=SIZE_D1B8) {
  perror("");
  printf("Error Reading %s File.\n", argv[1]);
  exit(3);
}
/* ************************************************
   * Chargement du nom complet du fichier dans le PI37 *
   ************************************************ */
 strncpy((char *)d1b8.label,"C:\\CXV301\\FICHIER\\NOMFICHIER",80);

/* Reecriture du fichier temporaire de TOM */
fseek(param,0,0);
bytes = fwrite((char *)&d1b8,1,SIZE_D1B8,param);
if (bytes!=SIZE_D1B8) {
  perror("");
  printf("Error Writing %s File. %d bytes written instead of %d\n",
        argv[1],bytes,SIZE_D1B8);
  exit(3);
}

fclose(param);
exit(0);
}
```

Procédures Utilisateurs

Connect:Express peut déclencher des procédures utilisateurs en début ou fin de transfert. La définition d'un fichier symbolique permet de préciser, dans le second écran, quelle commande déclencher en début et fin de transfert, en émission et en réception. L'écran ci dessous en donne l'exemple :

```
C:E/UNIX 145-1  -------------------- FILES DIRECTORY ---------------------- tom1
OPTION ===>

SYMBOLIC NAME        :             FICHIER     DEFINITION : D DIRECTION : R

TRANSMISSION :
START EXIT ............ : ............
START COMMAND ......... : CMDDEBT.....
END EXIT ............... : ............
END COMMAND ............ : CMDENDT.....

RECEPTION :
START EXIT ............ : ............
START COMMAND ......... : CMDDEBR.....
END EXIT ............... : ............
END COMMAND ............ : CMDENDR.....

DO YOU WANT TO GO ON ?
OPTION : VIEW                                  UPD : 20040601112010 C:E 141-3
-ENTER- NEXT FIELD                    -F3- CANCEL                    -F8-
COMPLETION
```

Le tableau suivant indique les composants fournis dans le répertoire /exit :

Module	Description
UEXERR	Procédure généralisée de gestion d'erreur
TRFOK	Exemple : affichage des paramètres
ROUTAGE	Exemple : routage vers destinataire (si # nom local)
UEXROUT	Procédure généralisée de transfert avec routage
ROUTPI62	Exemple : routage vers recepteur (si # nom local)

Les procédures utilisateurs doivent être copiées dans le répertoire /exit.
La commande est lancée par Connect:Express avec un ensemble de paramètres numérotés de 1 à 24. Connect :Express continue le traitement après avoir lancé la commande, sans en attendre le résultat .

Les paramètres passés à la commande utilisateur sont les suivants :

Paramètre	Contenu
$1	Numéro de requête
$2	Nom symbolique de fichier
$3	Nom symbolique de partenaire
$4	Nom physique de fichier
$5	Sens de transfert
$6	Code retour du système (SRC)
$7	Code retour de Connect:Express (TRC)
$8	Code retour protocolaire (PRC)
$9	Pi99 reçu
$10	Pi99 envoyé

$11	Origine du transfert
$12	Destination du transfert
$13	Nom local
$14	Label
$15	Emetteur du transfert
$16	Recepteur du transfert
$17	Date de soumission de la requête
$18	Heure de soumission de la requête
$19	Etat du transfert
$20	Quantième (Date Julienne de la requête)
$21	Nombre d'enregistrement transmis
$22	Nombre de K octets transmis
$23	Date de fin de transfert
$24	Heure de fin de transfert

Exemple Type de Traitement par Procédure Intégrée

La procédure ci dessous affiche les paramètres reçus en entrée :

```
#
#  SHELL Command TRFOK
#
REQ=”$1”
FIC=”$2”
PART=”$3”
NOM_PHY=”$4”
SENS=”$5”
SRC=”$6”
TRC=”$7”
PRC==”$8”
PI99R=”$9”
shift
PI99S=”$9”
shift
ORG=”$9”
shift
DST=”$9”
shift
LOC=”$9”
shift
LAB=”$9”
shift
PI61=”$9”
shift
PI62=”$9”
shift
RSD=”$9”
shift
RST=”$9”
shift
ETA=”$9”
shift
QQQ=”$9”
shift
NREC=”$9”
shift
KBYT=”$9”
Shift
RED=”$9”
Shift
RET=”$9”
```

```
echo "TRANFER TERMINATED WITHOUT PROBLEM"
echo "REQUEST               $REQ"
echo "FILE NAME             $FIC"
echo "PARTNER NAME          $PART"
echo "PHYSICAL NAME         $NOM_PHY"
echo "TRANSFER DIRECTION    $SENS"
echo "SYSTEM RETURN CODE    $SRC"
echo "TOM RETURN CODE       $TRC"
echo "PROTOCOL RETURN CODE $PRC"
echo "PI99 RECEIVED         $PI99R"
echo "PI99 SENT             $PI99S"
echo "TRANSFER ORIGIN       $ORG"
echo "TRANSFER DESTINATION $DST"
echo "LOCAL NAME            $LOC"
echo "LABEL                 $LAB"
echo "USER SENDER   (PI61) $PI61"
echo "USER RECEIVER (PI62) $PI62"
echo “REQUEST START DATE    $RSD”
echo “REQUEST START TIME    $RST”
echo “TRANSFER STATE        $ETA”
echo “JULIAN DATE           $QQQ”
echo “NUMBER OF RECORDS     $NREC”
echo “K. BYTES              $KBYT”
echo “REQUEST END DATE      $RED”
echo “REQUEST END TIME      $RET”
```

Annexe 1 - Fichier d0b8z20.h

La compilation du programme utilisateur devra inclure l'option –L $TOM_DIR/itom. La variable d'environnement correspond au répertoire racine du moniteur (par exemple /home/tom1).

La structure d0b8z20.h comprend en tête les sous structures et se termine par la structure globale, en sens inverse de la structure représentée ci dessous :

- *ZREQ_tom* Paramètres de la demande = En-tête **+ Demande**
  - *uni_sci* Demande = **Fichier** // **Partenaire** // **Requête de transfert** // **Enregistrement RENC**
    - *fichier :* Description d'un Fichier
    - *partenaire* : Description d'un Partenaire
    - *st_sci* : Paramètres d'une requête de transfert
    - *s_renc* : Enregistrement du fichier RENC = Etat du traitement **+ Paramètres du transfert**
      - *st_trf* : Paramètres du transfert = **Paramètres PeSIT + Paramètres réseau**
        - *s_pi* : Paramètres PeSIT
        - Paramètres réseau : *s_x25* : Paramètres X25 **//** *s_tcp* : Paramètres TCP/IP

```
/* Network Structures */
struct s_x25_param {
  char appelant[15];      /* Calling DTE Number             */
  char appele[15];        /* Called DTE Number              */
  char port[1];           /* Device Name                    */
  char applid[8];         /* Routing List(IBM)/LAID (NCR)   */
  char command[1];        /* Error Command                  */
  unsigned char loc_fac[8];  /* Facilities Field                */
  unsigned char udf[4];      /* User Data Field                 */
  int nrc;                   /* Network Return Code             */
  short lg_appelant;         /* Calling DTE Number Length       */
  short lg_appele;           /* Called DTE Number Length        */
  short lg_applid;           /* Routing List /LAID Length       */
  short loc_fac_lg;          /* Facility Field Length           */
  short udf_lg;              /* User Data Field Length          */
  unsigned char cause[2];    /* TRANSPAC Cause                  */
  unsigned char diagnostic[2]; /* TRANSPAC Diagnostic           */
};

#define S_X25 sizeof(struct s_x25_param)
struct s_tcp_param {
  char port[5];              /* Port Service                    */
  char adresse[15];          /* Partner Internet Addresse       */
  char host[32];             /* Host Name                       */
  char command[1];           /* Error Command                   */
  int nrc;                   /*  Network Return Code          */
  char filler[S_X25 - 53 - sizeof(int)];
};

#define S_TCP sizeof(struct s_tcp_param)
```

```
struct s_pi {
  char diag[3];      /* Diagnostic                    pi2  */
  char ident[8];     /* Identity of requester         pi3  */
  char idser[8];     /* Identity of server      pi4  */
  char cac[8];       /* Access control                pi5  */
  char ver[1];       /* Number of Protocol version    pi6  */
  char opo[3];       /* Synchro option                pi7  */
  char tyf[2];       /* File Type               pi11 */
  char nof[8];       /* Symbolic filename       pi12 */
  char idt[3];       /* Transfer identificator        pi13 */
  char atd[1];       /* Attributs asked               pi14 */
  char trr[1];       /* Restart flag            pi15 */
  char cod[1];       /* Data code                     pi16 */
  char prt[1];       /* Priority of transfert         pi17 */
  char por[3];       /* Point of restart              pi18 */
  char cft[1];       /* End of transfer code          pi19 */
  char nps[3];       /* Number of synchronization point    pi20 */
  char cpr[2];       /* Compression                   pi21 */
  char tac[1];       /* Access type             pi22 */
  char res[1];       /* Resynchronization             pi23 */
  char mlt[2];       /* Multi-fpdu length             pi25 */
  char nb_oct[4];    /* Number of bytes               pi27 */
  char nb_art[4];    /* Number of articles      pi28 */
  char far[1];       /* Format of article             pi31 */
  char loa[2];       /* Length of article             pi32 */
  char orf[1];       /* File organization             pi33 */
  char label[80];    /* Label du Fichier              pi37 */
  char vur[1];       /* Unit of space allocation pi41 */
  char vme[4];       /* Maximum space allocation      pi42 */
  char dhc[12];      /* Unused                  pi51 */
  char dhd[12];      /* Unused                  pi52 */
  char nom_phy[44];  /* File physical name                 */
  char oc5[1];       /* Identity of receiver               */
  char oc6[1];       /* Identity of transmitter            */
  char s_trace[1];   /* Flag = Y if request activation of trace  */
                     /* Flag = N if request deactivation of trace*/
  char no_req[8];    /* Request number                     */
  char typ_req[1];   /* Request Type (Normal or Inquiry)        */
  char typ_part[1];  /* Partner type (T or O)              */
  char userid[8];    /* Requester Userid                   */
  char format[2];    /* Record Format (Text, Binary, Fixed,     */
                     /* Variable Undefined) TF TV BF BU    */
  char etat[1];      /* Transfer State                     */
                     /* 0 : Waiting ('A')                  */
                     /* 1 : Connected                      */
                     /* 2 : Selected                       */
                     /* 3 : Opened                         */
                     /* 4 : Suspended                      */
                     /* 5 : Transfered                     */
                     /* 6 : Deselected                     */
  char s_dsname[44]; /* Sender Physical Name                    */
  char r_dsname[44]; /* Receiver Physical Name                  */
  char multiart[1];  /* Flag MultiArticle (Y/N)                 */
  char crc[1];       /* Crc Flag (Y/N)                          */
  char s_no_req[4];  /* Sen. Req. Number (Bin)                  */
  char s_req_dat[4]; /* Sen. Req. Date (Bin)                    */
  char s_group[8];   /* Sender User Group                       */
  char s_userid[8];  /* Sender User Id                          */
  char s_old_psw[8]; /* Sender Old Password                     */
  char s_new_psw[8]; /* Sender New Password                     */
  char rem_trc[4];   /* Remote TRC                              */
  char translat[1];  /* Translation Table                       */
```

```
  short trc;         /* TOM Return Code                    */
  short src;         /* System Return Code                 */
  short lg_ident;    /* Length of field 'ident'            */
  short lg_idser;    /* Length of field 'idser'            */
  short lg_fic;      /* Length of field 'nof'              */
  short support;     /* Type of link                       */
  int ack_pos;       /* Offset of last acquitted synchro point  */
  char num_prot[1];  /* Protocol Number (1:Etebac3, 3:PeSIT)    */
  char password[8];  /* Password of Partner                */
  char api[88];      /* API Field                          */
  char tsm[3];       /* Type/Structure/Mode FTP            */
  char alloc[1];     /* Allocation Flag Y/N                */
  char rule[1];      /* Allocation Rule 0,1,2              */
  char stou[1];      /* Sore Unique Flag Y/N FTP           */
  char fa[1];        /* flag File agent Y/N                */
  char ack_pos_lfs[8]; /* 64 bits (Large file support)      */
  char nb_oct_lfs[8];  /* 64 bits (Large file support)      */
  char s_pi99_254[254]; /* Pi99 sent                         */
  char r_pi99_254[254]; /* Pi99 received                     */
  char user_org[8];     /* User Origin              pi3 bis  */
  char user_dst[8];     /* User Destination         pi4 bis  */
  short lg_user_org;    /* Length of field 'origin'          */
  short lg_user_dst;    /* Length of field 'destination'     */
  char user_snd[24];    /* User Sender              pi61     */
  char user_rcv[24];    /* User Receiver            pi62     */
  short lg_user_snd;    /* Length of field 'sender'          */
  short lg_user_rcv;    /* Length of field 'receiver'        */
  char quant[3];        /* Julian Date                       */
  char n_rep[2];        /* Number of retries allowed         */
  char int_sess[2];     /* Interval Session Timer            */
  char int_trans[2];    /* Interval Transfer Timer           */
  char notif[1];        /* Notification                      */
  char sslparmid[8];    /* SSL PARLMID                       */
  char sslrc[8];        /* Openssl error code                */
  char ssl_used[1];     /* SSL used                          */
  char flag_rout[1];    /* Flag rout                         */
  char filler[105];     /* For Future Use                    */
};

struct st_trf {
  struct s_pi pi;
  union {
    struct s_x25_param x25_param;
    struct s_tcp_param tcp_param;
  } u_netw_param;
};

/* Definitions facilitant l'acces aux zones X25 et TCPIP */
#define X25   u_netw_param.x25_param
#define TCPIP u_netw_param.tcp_param

typedef struct s_renc  {  /* Record of RENC file.          */
  char typ_dem[1];      /* 'D' Requester  'S' Server       */
  char nom_fic[8];      /* Logical name of file.           */
  char dat_sou[8];      /* Date of soumission, YYYYMMDD.   */
  char heu_sou[6];      /* Time of soumission, HHMMSS.     */
  char sen_tra[1];      /* Direction of transfer (T-Transmit,  */
                        /* R-Receive)                      */
  char eta_tra[1];      /* State of transfer :             */
                        /*   C - in progress               */
                        /*   A - awaiting selection        */
```

```
                        /*   D - defered                          */
                        /*   E - ended                           */
                        /*   O - abnormally ended                 */
                        /*   J - automatic restart                */
                        /*   K - awaiting restart                */
  char dat_deb[8];      /* Date of start of transfer (YYYYMMDD) */
  char heu_deb[6];      /* Time of start of transfer (HHMMSS)   */
  char dat_fin[8];      /* Date of end of transfer   (YYYYMMDD) */
  char heu_fin[6];      /* Time of end of transfer   (HHMMSS)   */
  int nb_repr;          /* Number of retries (initialised to 0) */
  char ori_com[1];      /* Origin of command:                   */
                        /*   S - TOM menu.                      */
                        /*   I - Application interface          */
                        /*   T - Batch                          */
  short ses;            /* Session Number which made the transfer*/
  int pid;              /* Process ID of STRF which transferred */
  char support_origin;  /* Origin support of request            */
  char filler[13];      /* For Future Use                       */
  struct st_trf trfpar; /* Structure whith transfer parameters  */
} s_renc;

#define SIZE_RENC sizeof(struct s_renc)

struct st_sci {
  char dire[1];                 /* Direction                 */
  char file[8];                 /* Symbolic file name        */
  char part[8];                 /* Symbolic partner name     */
  char dsnam[44];               /* Dsname                    */
  char prty[1];                 /* Priority                  */
  char dat[8];                  /* Date                      */
  char hour[6];                 /* Hour                      */
  char lnk[1];                  /* Link type                 */
  char udf[44];                 /* User data file            */
  char typ[1];                  /* Request type              */
  char sta[1];                  /* State of Request          */
  char dpcsid[8];               /* Dpcsid for Alias          */
  char dpcpsw[8];               /* Dpcpsw for Alias          */
  char format[2];               /* Record Format TF TV BF BU */
  char lrecl[5];                /* Record Length             */
  char api[88];                 /* Api Field                 */
  char tsm[3];                  /* Type/Structure/Mode FTP   */
  char stou[1];                 /* Store Unique FTP          */
  char fa[1];                   /* flag File agent Y/N       */
  char label[80];               /* Label du Fichier          */
  char s_pi99_254[254];         /* pi99 254 characters       */
  char user_org[8];             /* User Origin               */
  char user_dst[8];             /* User Destination          */
  char user_snd[24];            /* User Sender               */
  char user_rcv[24];            /* User Receiver             */
  char quant_aa[2];             /* AA for Julian Date        */
  char quant[3];                /* Julian Date               */
  char notif[1];                /* Notification              */
  char filler[SIZE_RENC - 643];
};


/*************************************************************/
/*               Structure for displaying partner             */
/*************************************************************/
struct partenaire {
  char nom_sym[8];   /* Partner Symbolic Name              */
  char passwd[8];    /* Password                           */
```

```
  char etat_init[1]; /* Initialization State                */
  char nature[1];    /* Partner Type (TOM or Compatible)    */
  char num_prot[1];  /* Protocol Number (1->ETB3, 3->PeSIT) */
  char tab_sess[1];  /* Session Table (1 -> 9)              */
  char port[1];      /* X25 Device                          */
  char nb_liai[2];   /* Number of Sessions                  */
  char typ_lia[1];   /* Type of Link (L, X, M)              */
  char num_rem[15];  /* X25 Remote Address                  */
  char num_loc[15];  /* X25 Local (Sub)Address              */
  char loc_fac[16];  /* Facilites                           */
  char udf[8];       /* User Data Filed                     */
  char upd_date[14]; /* Date of Last Updating               */
  char userid[8];    /* Userid who updates                  */
  char dpcsid[8];    /* DPCSID alias                        */
  char dpcpsw[8];    /* DPCPSW alias                        */
  char tcp_host[32]; /* Host Name                           */
  char tcp_addr[15]; /* Host Adresse                        */
  char tcp_port[5];  /* Host Port                           */
  char ftpfile[8];   /* FTP Default File                    */
  char nb_liai_in[2];  /* Number of Sessions IN    15/01/01 */
  char nb_liai_out[2]; /* Number of Sessions OUT   15/01/01 */
  char nb_repr[2];     /* Number of Retries                 */
  char int_sess[2];    /* Interval Session Timer            */
  char int_trans[2];   /* Interval Transfer Timer           */
  char sslparmid[8];   /* SSL PARMID                        */
  char filler[62];     /* for future use                    */
};

/*****************************************************************/
/*               Structure for displaying file                   */
/*****************************************************************/
struct fichier {
  char nom_sym[8];    /* Symbolic File Name          */
  char etat_init[1];  /* Initialization State        */
  char direction[1];  /* Direction                   */
  char recepteur[8];  /* Receiver Partner            */
  char emetteur[8];   /* Sender Partner              */
  char priorite[1];   /* Priority                    */
  char typ_def[1];    /* Definition Dyn/Fix          */
  char present[1];    /* Presentation Table 1->9     */
  char nom_phy[44];   /* Dsname                      */
  char format[2];     /* TV, TF, BF, BU              */
  char record[5];     /* Record Length               */
  char exit_de[12];   /* Exit/Comm Transm.           */
  char comm_de[12];
  char exit_fe[12];
  char comm_fe[12];
  char exit_dr[12];   /* Exit/Comm Reception         */
  char comm_dr[12];
  char exit_fr[12];
  char comm_fr[12];
  char upd_date[14];  /* Date of Last Updating       */
  char userid[8];     /* Userid who updates          */
  char remotedsn[44]; /* Remote DSN                  */
  char tsm[3];        /* Type/Structure/Mode         */
  char tab[1];        /* Optionnal Table             */
  char rule[1];       /* Allocation Rule 0/1/2       */
  char alloc[1];      /* Allocation Flag Y/N         */
  char stou[1];       /* Store Unique Flag Y/N FTP   */
  char fa[1];         /* Flag File agent Y/N         */
  char notif[1];      /* Notification                */
```

```
  char filler[5];     /* For Future Use               */
};

/*****************************************************************/
/* Union                                                         */
/*****************************************************************/
union uni_sci {
  struct st_sci zreq_tom_sci;
  struct s_renc zreq_tom_renc;
  struct partenaire zreq_tom_part;
  struct fichier zreq_tom_fic;
};

struct ZREQ_TOM {
  char zreq_tom_name[4];           /* Monitor name           */
  char zreq_tom_func[1];           /* Function type          */
  char zreq_tom_tabn[1];           /* Request type              */
  char zreq_tom_reqn[8];           /* Request number            */
  char zreq_tom_rtcf[1];           /* Tom return code           */
  char zreq_tom_rscf[3];           /* Reason return code        */
  union uni_sci uni;
};
```

Annexe 2 - Fichier d1b8ruex.h

Le fichier d1b8ruex.h comprend en tête la description du Pi99 utilisé avec les partenaires de type TOM (st_pi99). Cette structure n'est valable que pour un usage spécifique entre moniteurs Connect :Express. Lorsque le type de partenaire est Other, le Pi99 est libre. La structure suivante (d1b8ruex) est utilisée pour les transferts PeSIT et FTP. La dernière structure (d1b8etb3)est passée à l'exit ETEBAC3.

```
/*****************************************************************/
/*                                                               */
/*                      D1B8RUEX.H FILE                          */
/*                                                               */
/*****************************************************************/
struct st_pi99 {
       /*--- VERSION IDENTIFIER ---*/
       char pesit_version[1];
       char pi99_version[1];
       /*--- REQUEST ELEMENTS ---*/
       char request_number[4];
       char request_date[4];
       char performance_flag[1];
       char type[1];
       char group12[2];
       /*--- RACF ELEMENTS ---*/
       char user_remote_id[8];
       char old_password[8];
       char new_password[8];
       /*--- ELEMENTS OF SENDER FILE ---*/
       char s_dsname[44];
       char s_blksize[2];
       char s_lrecl[2];
       char s_recfm[1];
       char s_dcb_format[1];
       char s_disp1[1];
       char s_unit[8];
       char s_volcount[1];
       char s_volser1[6];
       char s_volser2[6];
       char s_volser3[6];
       char s_volser4[6];
       char s_volser5[6];
       char group21[1];
       char group31[1];
       char tape_only[6];
       char group42[2];
       /*--- ELEMENT OF RECEIVER FILE ---*/
       char r_dsname[44];
       char r_blksize[2];
       char r_nbdir[2];
       char transmit_retpd[2];
       char r_lrecl[2];
       char r_recfm[1];
       char r_dcb_format[1];
       char r_disp1[1];
       char r_disp2[1];
       char r_disp3[1];
       char expdt[3];
```

```
       char r_unit[8];
       char r_volcount[1];
       char r_volser1[6];
       char r_volser2[6];
       char r_volser3[6];
       char r_volser4[6];
       char r_volser5[6];
       char device_dependent_seg[14];
       char group52[2];
       char last_byte[1];
};     /* 254 taille max */
/*****************************************************************/
/*                                                               */
/*  Temporary file structure given during exit execution         */
/*                                                               */
/*****************************************************************/
struct d1b8ruex {
  char subsys[4];       /* Monitor name                          */
  char date_launch[14];
  char request[8];      /* Local Request Number                  */
  char file[8];         /* Symbolic filename                     */
  char partner[8];      /* Symbolic partner name                 */
  char dsn[44];         /* Dsname                                */
  char direction[1];    /* Transfer direction                    */
  char type[1];         /* Exit type 'I'(nit),'E'(nd)or'F'(ailed)*/
  char trfid[8];        /* Identification of transfer            */
  char local[8];        /* Local symbolic name                   */
  char src[4];          /* System Return Code                    */
  char trc[4];          /* Tom Return Code                       */
  char diag[4];         /* Protocol Return Code                  */
  char requester[8];    /* Requester Symbolic Name               */
  char server[8];       /* Server Symbolic Name                  */
  char tyf[5];          /* File Type                             */
  char priority[1];     /* Priority                              */
  char cod[1];          /* Data Code (0:Ascii, 1:Ebcdic,2:Binary)*/
  char label[80];       /* File Label                            */
  char dhc[12];         /* File Creation Date and Time           */
  char dhd[12];         /* File Extraction Date and Time         */
  union {
    struct st_pi99 pi99_new;
    char pi99_old[64];
  } pi99;               /* Pi 99 (Depends on Protocol Version)   */
                        /* #1:First 64 bytes with 'TOM' in EBCDIC*/
                        /* #2 : 1st Byte : 0X02 2nd Byte : 0X01  */
  char begining[14];    /* Beginning of transfer date/time       */
  char ending[14];      /* Ending of transfer date/time          */
  char duration[8];     /* Transfer duration. Unused             */
  char kbytes[7];       /* Kbytes Transfered                     */
  char records[10];     /* Records Transfered                    */
  char lrecl[5];        /* Record Length                         */
  char recfm[2];        /* Record Format (TV, TF, BF, BU)        */
  char kbytes_lfs[12];     /* LARGE FILE SUPPORT                  */
  char s_pi99_254[254]; /* Sender pi99 on 254                    */
  char r_pi99_254[254]; /* Receiver pi99 on 254                  */
  char user_org[8];     /* User Origin                           */
  char user_dst[8];     /* User Destination                      */
  char user_snd[24];    /* User Sender    (pi61)                 */
  char user_rcv[24];    /* User Receiver  (pi62)                 */
};

/*****************************************************************/
/*                                                               */
```

```
/*  Temporary file structure given during ETEBAC3 exit execution */
/*                                                               */
/*****************************************************************/
struct d1b8etb3 {
  char client_or_bank[1]; /*   0 */  /* 'C' = client, 'B' = Bank */
  char msg_type[3];       /*   1 */  /* 'DEB' 'FIN'              */
  char msg_orig[1];       /*   4 */  /*  T = set by TOM          */
                                     /*  C = set by Client       */
                                     /*  B = set by Bank         */
  char param_card[80];    /*   5 */
  char dial_number[15];   /*  85 */
  char user_data[8];      /* 100 */
  char oknok[20];         /* 108 */
  char user_retcode[4];   /* 128 */
  char tom_retcode[4];    /* 132 */
  char request[8];        /* 138 */
  char physical_name[44]; /* 144 */
  char client[8];         /* 188 */
  char bank[8];           /* 196 */
  char filler[52];        /* 204 */
};                        /* size = 256 */
```

Annexe 3 - Procédures de Compilation

La compilation d'un programme utilisateur qui appelle la fonction L0B8Z20 doit inclure l'option –L $TOM_DIR/itom. La variable d'environnement correspond au répertoire racine du moniteur (par exemple /home/tom1).

Les deux exemples ci dessous représentent la procédure de compilation du programme et la procédure de make de l'exécutable.

```
#    Cette procedure permet de compiler un fichier source en module
OBJET.
echo ""
echo "Compilation du Programme Source test_api.c"
echo ""
make -f makefile.test_api
if [ $? = "0" ]
then echo "No error....."
else
     echo "Error....."
     exit 1
fi
```

```
#    Fichier de make pour test_api

#    Pour la Fin
fin        :    test_api
           @ls -l *.lis >> trace.lis
           @cat trace.lis

test_api   : test_api.o
             cc -g -o test_api test_api.o -L $(TOM_DIR)/itom -lc -
litom \
                      2> ld_test_api.lis

test_api.o :  test_api.c
             cc -c test_api.c 2>cc_test_api.lis
```