Introduction

Depuis presque 40 ans, EASYCOM accompagne PC Soft en assurant un outil complet et puissant pour l'accès natif AS400 sur l'ensemble de ses outils de développement.

Easycom For Windev 2026 et **Easycom For WebDev 2026** viennent compléter une longue série d'évolutions et d'améliorations.

Cette documentation est commune aux deux environnements mais l'utilisation d'Easycom requiert une licence propre à chaque AGL et il faut bien sûr disposer de la version WinDev ou WebDev correspondante.

Pour simplifier le texte dans la suite de cette documentation, le terme « WINDEV » sera utilisé. Les spécificités WebDev sont précisées dans des chapitres distincts.

Easycom vous offre un accès global aux ressources de l'AS/400 :

- accès natif aux données par les fonctions classiques (`HLitPremier`, etc...)
- accès SQL par les fonctions `HExecuteRequete`,
- appel de commandes simples ou avec retour de paramètres,
- accès aux programmes et aux data queues (représentés par une description de fichier),

Pour ce qui est de l'accès aux fichiers ou l'utilisation de requêtes SQL, le principe de base est la plus grande compatibilité avec les principes et les fonctions HyperFile et le W-Langage. Vous manipulez un fichier AS400 de la même manière et avec le même code qu'un fichier Hyperfile avec cependant quelques restrictions dues aux spécificités de l'AS/400, certaines fonctions ou paramètres optionnels ne sont pas supportés.

Le mode SQL permet l'utilisation de requêtes complexes, préparées, avec en plus les fonctionnalités de mise à jour ou d'indexation.

Les transactions sont également gérées.

Selon la nature du projet, il est possible de travailler sur des tables AS400 existantes, en importants les descriptions de fichiers dans une analyse, ou bien d'exporter une analyse vers l'AS400 en créant les fichiers, avec ou sans les éventuelles données.

Les programmes et data queues se manipulent par l'intermédiaire de fichiers "image" qui les décrivent et qui doivent être importés.

Vous disposez par ailleurs de fonctions spécifiques pour changer d'utilisateur, envoyer des commandes, appeler des programmes, récupérer des paramètres ou des messages d'erreur ou affiner certaines options :

- Lancement d'une commande sur AS/400 : AsExec

- Lancement d'une commande AS/400 avec retour de valeurs : AsAppelRTV
- Récupération du résultat d'une commande : AsResultatRTV
- Appel d'un programme AS/400 : AsLanceRPC
- Appel d'un programme AS/400 nouvelle interface: ASAppelPgm
- Adoption des droits d'un autre utilisateur : AsUtilisateur
- Modifier les propriétés et paramètres : ASPropriete
- Détail des erreurs (type CPF) : AsErreurAide et AsErreurDonnée

Installation

Configuration requise

Serveur

- Tous les modèles AS/400 depuis la série B.
- Easycom est compatible avec l'OS/400 jusqu'à la V7R5.
  La V5R3 est la version minimum recommandée.
- Connexion vers des PC par réseaux TCP/IP.

Client

- Protocole TCP/IP.
- Système d'exploitation : toutes les versions de Windows supportées par Microsoft.

Logiciel

- Outil de développement : WinDev 2026 - WebDev 2026

Un profil de type QSECOFR est nécessaire pour l'installation de la partie serveur.

Conditions préalables

Pour l'installation, le profil utilisateur de l'AS/400 à utiliser doit être du type 'QSECOFR' (il est possible de l'indiquer lors de l'installation).

Avant d'installer la partie "Serveur", il faut connaître les caractéristiques de la connexion :

- Type de connexion : TCP/IP
- Service FTP démarré, uniquement pour l'installation. Pour démarrer FTP utiliser STRTCPSVR (*FTP). Vérifiez éventuellement la connexion avec le client FTP de Windows.
- Nom de l'AS/400 ou son adresse IP.

Installation du serveur Easycom sur l'AS400

Le serveur EASYCOM est constitué d'un ensemble d'objets (programmes, commandes et fichiers) regroupés dans une seule bibliothèque.

Par défaut il s'agit de la bibliothèque **'EASYCOM'** mais il est possible de spécifier un autre nom de bibliothèque ou de combiner plusieurs installations. On suppose dans la suite que l'installation est faite dans EASYCOM.

Le serveur EASYCOM doit être installé une fois sur l'AS/400. Son **installation se fait à partir d'un PC** sous Windows, connecté à l'AS/400 par TCP/IP.

Le serveur EASYCOM assure une compatibilité de toutes les versions clientes.

TCP/IP

Sur l'AS/400, TCP/IP doit être installé, configuré et démarré (voir les commandes CFGTCP et STRTCP).

Serveur FTP sur l'AS/400

L'installation par TCP/IP utilise FTP.

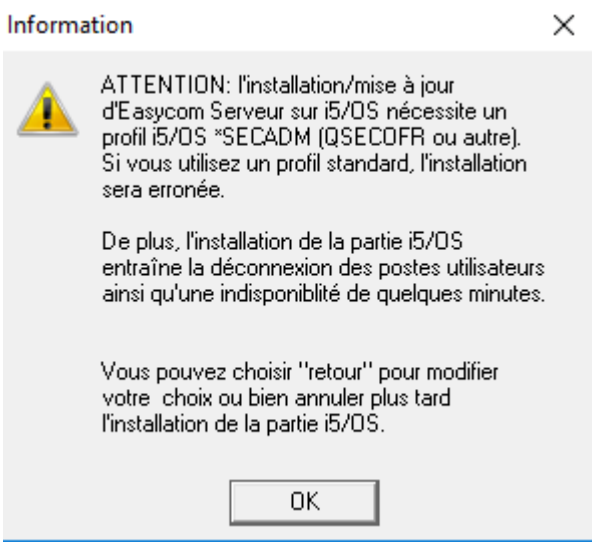
Le serveur FTP est nécessaire pour l'installation du serveur sur l'AS/400.

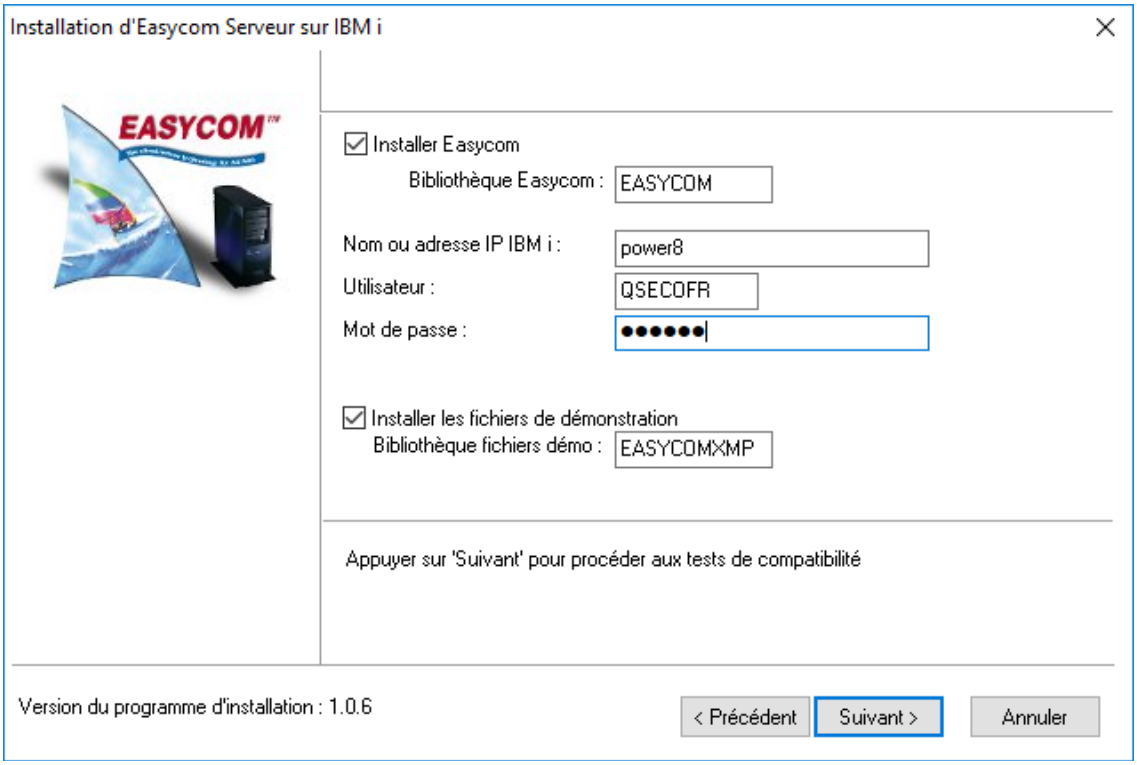
Il faut pour cela démarrer le service FTP sur l'AS/400 par la commande `STRTCPSVR SERVER(*FTP')`.

Il n'est pas nécessaire pour le fonctionnement normal d'EASYCOM.

Profil à utiliser pour l'installation

L'installation nécessite un profil de type QSECOFR.





Déroulement de l'installation du serveur

- **Vous devez confirmer le nom de la bibliothèque d'installation sur l'AS/400.**

  Sauf dans le cas d'une installation pour tester une nouvelle version, ou d'un conflit de noms, il est déconseillé de changer le nom de cette bibliothèque.

- **Installation des fichiers de démonstration :**

  Ces fichiers de démonstration sont nécessaires pour le bon fonctionnement des programmes de test et de démonstration installés sur le poste client avec l'environnement de développement.

EASYCOM communique par le protocole TCP/IP entre les postes clients et le serveur AS/400.
C'est FTP qui est utilisé pour télécharger la bibliothèque du serveur EASYCOM.

- **Il faut donner le nom ou l'adresse IP de l'AS/400 sur lequel l'installation sera faite.**
- **Il est également nécessaire d'entrer un nom d'utilisateur et un mot de passe.**

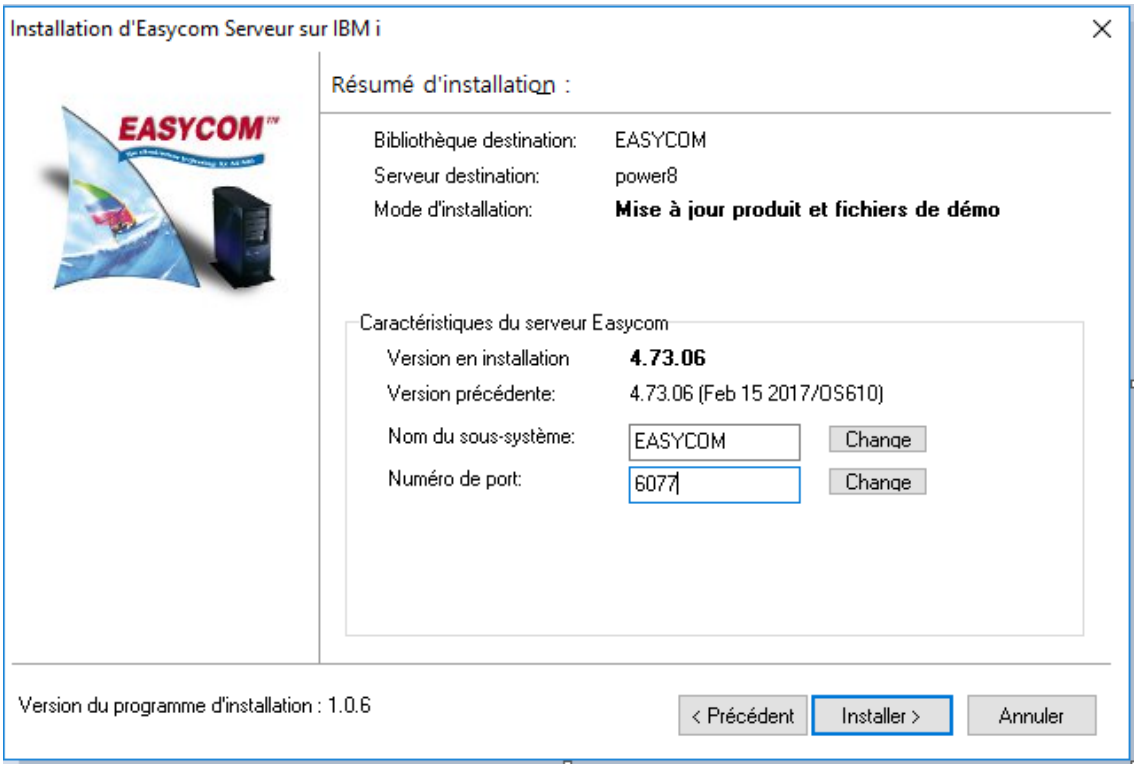
  Il est déconseillé d'utiliser un autre utilisateur que QSECOFR : l'installation ne pourra fonctionner correctement qu'avec un profil ayant les autorités spéciales *ALLOBJ et *SECADM.
  Certains objets de la bibliothèque EASYCOM sont configurés pour avoir QSECOFR comme propriétaire.
  L'objet EASYCOMD (*PGM) doit tourner avec les droits de QSECOFR.

  Si l'installation du serveur n'est pas faite par QSECOFR, l'auto-configuration risque de ne pas s'effectuer intégralement, et les premiers démarrages peuvent être difficiles.

Test de la configuration actuelle

Après validation des informations de connexion et de bibliothèque, le programme d'installation procède à des tests de routine : existence d'une version précédente, droits suffisants.

Cela aboutit à l'écran de confirmation d'installation suivant :



Il est rappelé sur l'écran : la bibliothèque de destination, le type d'installation (nouvelle ou mise à jour), la version en cours d'installation.

En cas de mise à jour, le numéro de version de la version actuelle est affiché.

Cet écran permet également de choisir ou modifier le nom du sous-système et le port TCP/IP utilisés dans cette bibliothèque (le principe étant un sous-système par bibliothèque).

Opérations effectuées sur l'AS/400

Création d'une bibliothèque EASYCOM et restauration de divers objets dans cette bibliothèque.

Opérations effectuées sur le PC

Création d'un répertoire Easycom (par défaut C:\Program Files (x86)\Easycom), de sous répertoires spécifiques et copie de divers fichiers.

Voir Configuration et administration du serveur EASYCOM .

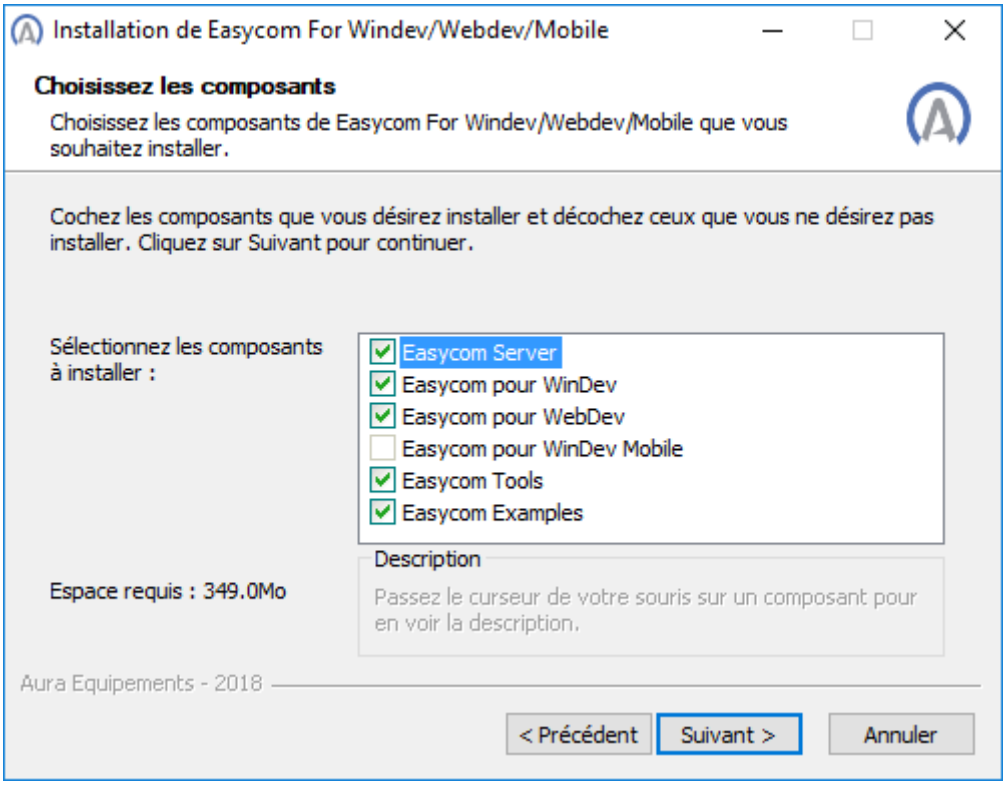
Installation d'Easycom For Windev sur un poste de développement

La procédure d'installation d'Easycom for Windev s'effectue à partir du programme d'installation téléchargé (à partir de votre espace client sur notre site web) sur un PC Windows.

Il vous sera également proposé de procéder à l'installation de la partie serveur sur l'AS/400 (**la partie AS/400 n'est à installer qu'une seule fois sur l'AS/400). Décochez la case si besoin**. Voir Partie : Installation du serveur Easycom sur l'AS400.

Le même programme d'installation est utilisé pour Windev et Webdev. Sélectionnez votre choix selon le produit installé (pour Windev et Webdev) sur votre PC.

Pour pouvoir procéder à l'installation, il faut qu'au moins un des 2 produits (Windev ou Webdev) soit déjà installé sur le PC.



Déroulement

Pendant l'installation les opérations suivantes seront effectuées sur le PC :

- Création d'un répertoire **"Program Files\EASYCOM".**
- Création d'un sous répertoire spécifique pour les exemples.
- Copie de divers fichiers

Par défaut les DLL de l'accès natif sont copiées dans C:\WinDev 31\Programmes\Framework\Win32x86 et C:\Windev31\Programmes\Framework\Win64x86 (pour les DLL 64 bits).

WinDev 2026 et/ou WebDev 2026 doivent évidemment être installés sur le poste de développement.

Remarque : L'installation d'Easycom For WinDev 2026 n'exerce aucune influence sur les programmes WinDev existants.

Versions antérieures

Les DLL Easycom communes du répertoire Windows sont compatibles avec les applications développées sous WinDev 2025 (30), 29, 28, 27 / 26 / 25 / 24 / 23 / 22 / 21/ 20 / 19 / 18 / 17 / 16 / 15 / 14 / 12 / 11/ 10 / 9 / 8 / 7.5 ou 5.5.

Il est également important de conserver et d'avoir accès aux anciennes DLL de chaque version. Le numéro de version apparaît dans le nom (eac2600as.dll pour WinDev 26).

Principaux fichiers

Sur le PC - poste d'un développeur

Dans le répertoire système de Windows :

easyco32.dll
easyco32.fr
eac32slf.dll
eac32slf.fr
eac32prc.dll
easycom.ini

Dans le répertoire de WinDev 2026 :

eac3100as.dll (DLL d'accès à l'AS/400)
eac3100as.fr (Affichage des Messages d'erreur et boîtes de dialogue en Français)
eac3100tr.dll (DLL de trace pc)

Dans le répertoire d'installation choisi :

eac32prc.exe (Constructeur DTAQ-RPC)
export_as400.exe (Constructeur de DDS)
eaccfg.exe (Easycom Configuration)

Sur le PC - poste du client final

Toutes les DLL nécessaires sont proposées par l'assistant de création d'installation de WinDev.

Sur le PC - poste d'un utilisateur

Dans le répertoire de l'application ou Windows :

Eac3100as.dll
Eac3100as.fr (facultatif, pour les messages d'erreur et boîtes de dialogue en français)

Remarques, en fonction des différentes plateformes :

Eac3100as.dll pour Windev 32 bits
Eac3100as64.dll pour Windev 64 bits
Eac3100as.so pour Linux

Sur l'AS/400

Voir installation du Serveur Easycom.

La bibliothèque "EASYCOM" contient la totalité des objets constituant le serveur EASYCOM.

Elle contient un grand nombre d'objets, programmes de configuration, exit programs, fichiers exemple...

Les fichiers de données susceptibles d'être modifiés :

YPROCHDR : Modifié par la description des procédures.
YPROCPARMS : Modifié par la description des procédures.
EACJOBD : Description par défauts des jobs Easycom
EACSESSION : Modifié par l'enregistrement des connexions.
AURA : Modifié par l'enregistrement des licences.

Installation d'un déploiement

Lors de la création de l'exécutable Windev, la DLL de l'Accès Natif AS/400 (**eac3100as.dll**) apparaît toujours dans les DLL obligatoires dès lors qu'un fichier de type AS/400 se trouve dans l'analyse ou qu'une fonction Easycom est utilisée dans le code.

Quel que soit le déploiement, seul le fichier **eac2900as.dll** est nécessaire.

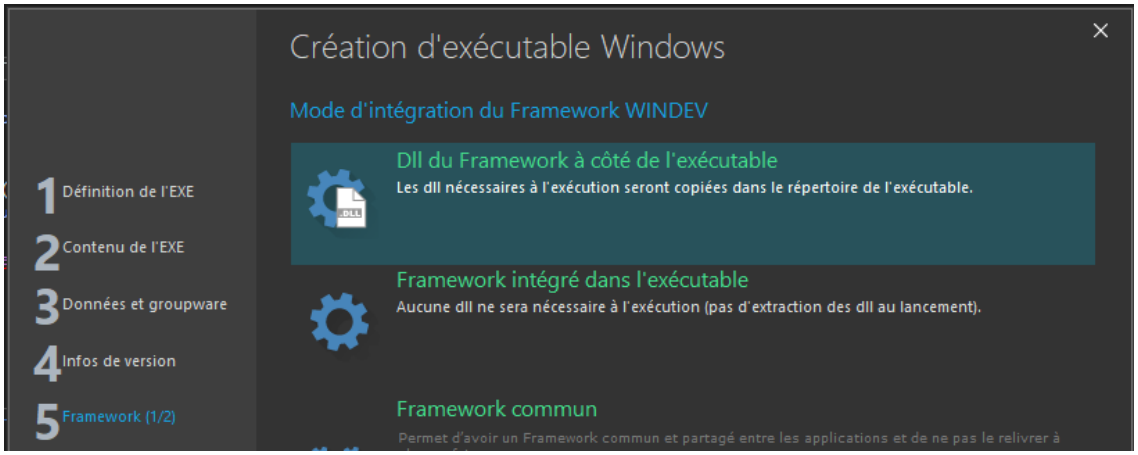
La suite concerne le déploiement d'une application WinDev.

Lors de la création du programme d'installation de l'application WinDev, **tous** les fichiers Easycom nécessaires sont proposés pour être automatiquement intégrés.

Il n'est donc pas nécessaire d'installer Easycom sur le poste de l'utilisateur final de l'application.

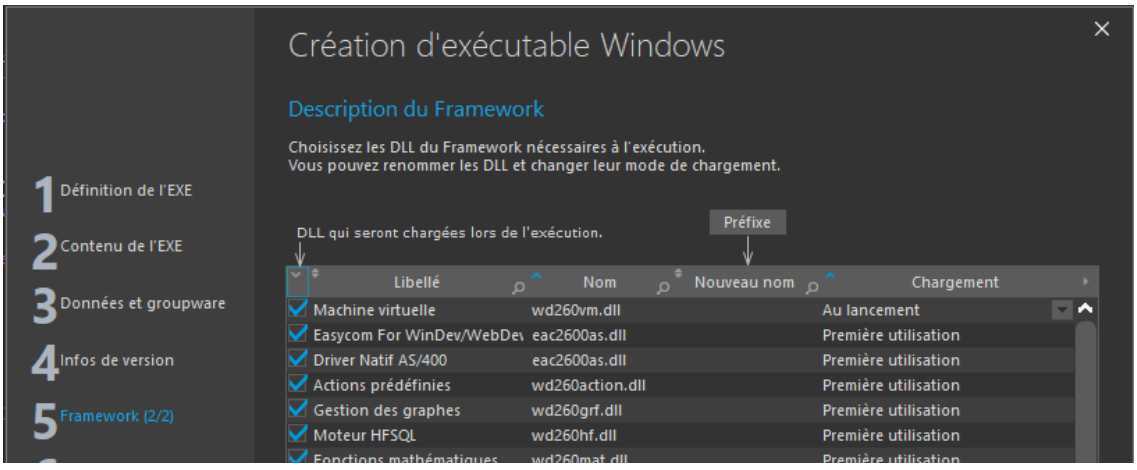
Fichiers proposés par l'assistant de création de l'installation :

Lors dela création de l'exécutable, il faut choisir "Dll du Framework à côté de l'exécutable" :



Et en choisissant cette option, vous verrez que seul le fichier ci-dessous est nécessaire et suffit pour le fonctionnement d'Easycom et de l'accès natif : **Eac3100as.dll**

Exemple avec Windev 26 :



Eventuellement, vous pouvez également inclure :

- un fichier easycom.ini, si vous souhaitez fixer une configuration particulière (adresse IP, cache...) activer les traces ou le Single Sign

- le programme Easycom Configuration :

Sélectionner dans le répertoire C:\Program Files\Easycom les fichiers EACCfg.exe et EACCfg.fr (pour les messages en français). Dans ce cas, easycom.ini n'est pas indispensable puisqu'il sera automatiquement créé par le programme Easycom Configuration.

Programmes et/ou data queues

Si le déploiement concerne un autre AS400 que le serveur de développement et si l'application utilise des procédures stockées ou des data queues, il faut copier les descriptions sur le nouvel AS400.

Vous pouvez copier directement le fichier YPROCHDR et YPROCPARMS de la bibliothèque EASYCOM si les deux AS/400 sont en réseau (ou sauvegarder et restaurer).

Vous pouvez utiliser le constructeur RPC-DTAQ pour sauvegarder les descriptions dans un fichier texte (extension .rpc) pour le restaurer ensuite sur le nouvel AS400 avec le même outil.

Déploiement sur un autre AS400

Le serveur Easycom doit bien sûr être installé et configuré et disposer d'une licence d'utilisation (Déploiement).

Le constructeur de DDS propose de créer un "script" d'installation avec un exécutable et les fichiers que l'on souhaite transférer.

Pour utiliser Easycom sur plusieurs AS400 il faut disposer des licences adéquates pour chaque AS400.

Icônes créées dans le menu "Démarrer"

Aide de Easycom For WinDev/WebDev 2026

Permet d'accéder à cette aide en ligne.

Configuration RPC - DTAQ

Permet d'accéder au Constructeur DTAQ – RPC.

Voir : Décrire les programmes natifs AS/400

Voir : Appel de procédure de programme de service

Easycom Configuration

Paramétrage, optimisation, vérification des versions, activation du Single-Sign-On et du fichier de trace.

Exemples

Permet d'accéder aux exemples d'Easycom For WinDev 2026.

Exportation de fichiers vers l'AS/400

Utilitaire de création des DDS et des fichiers à partir d'une analyse WinDev.

Installation du serveur Easycom

Permet d'accéder au programme d'installation de la partie AS/400 : EASYCOM Serveur.

Voir : Installation du serveur Easycom

Mises à jour d'Easycom For WinDev

Permet d'accéder directement à la espace client sur notre site web, avec votre code d'accès.

Cette page permet de télécharger la toute dernière version du produit.

Support Technique d'Easycom For WinDev

Pour les demandes de support il faut créer un ticket via le site internet du support : http://support.easycom-ra.com.
Il faut s'enregistrer la première fois, afin de recevoir un mot de passe.

Désinstallation

Pour désinstaller la partie cliente, passer par le panneau de configuration (Menu Démarrer, Paramètres) et la commande Ajout/Suppression de programmes.

Dans la liste, choisissez "Easycom For WinDev 2026" et cliquez sur le bouton "Ajouter/Supprimer".

Sur l'AS/400, pour désinstaller EASYCOM, il vous suffit de supprimer la bibliothèque.

Pour ce faire, arrêtez tous les jobs du sous-système EASYCOM, puis le sous-système lui-même :

```
ENDSBS EASYCOM * IMMED
```

de façon à ce que plus aucun objet n'utilise la bibliothèque, avant de la supprimer.

Pour contrôler, utilisez la commande :

```
WRKOBJLCK QSYS/EASYCOM OBJTYPE(*LIB)
```

Installation avancée

Cohabitation de plusieurs versions d'Easycom for Windev

1. La partie PC est à installer sur chaque poste développeur.

Les parties PC d'Easycom For WinDev sont différentes selon les versions de WinDev.

- Easycom For WinDev 2026 compatible avec WinDev 2026
- Easycom For WinDev 28 compatible avec WinDev 28

Il en est de même pour Easycom For WebDev.

2. La partie EASYCOM serveur sur l'AS/400 n'est à installer qu'une seule fois.

En effet, EASYCOM serveur est compatible avec tous les produits clients Easycom (PHP, .NET…), sous reserve d'installer la version Serveur la plus récente.

Cette version fonctionne avec toutes les versions antérieures d'Easycom For Windev (30, 29, 28, 27, 26, 25, 24, 23, 22…).

Easycom Serveur est installé par défaut dans la bibliothèque EASYCOM, sur le port 6077.

Cohabitation de plusieurs versions d'Easycom Serveur

Un serveur Easycom repose sur les propriétés suivantes :

- Une bibliothèque contenant tous les objets (nom par défaut : EASYCOM)
- Un sous-système (nom par défaut : EASYCOM)
- Un port tcp (valeur par défaut : 6077)

Pour mettre ne place un serveur supplémentaire sur un système, il est nécessaire d'installer Easycom dans une nouvelle bibliothèque. Il faut également attribuer un nom de sous-système ainsi qu'un port TCP.

Pour cela, procéder par les étapes suivantes :

1. Installer le serveur Easycom à l'aide de l'installation classique. Fournir simplement un nom de bibliothèque nouveau (par exemple EASYCOM2).
2. Une fois l'installation terminée, il faut créer le nouveau sous-système et lui attribuer un port TCP. Utilisez CFGEACSBS à cette fin.

Exemple :

Pour installer un nouveau serveur Easycom dans la bibliothèque EASYCOM2, sous-système EASYCOM2 avec le port 6078, utiliser les commandes suivantes :

ADDLIBLE EASYCOM2

CFGEACSBS SBS(EASYCOM2) PORT(6078)

Sur les postes clients, vous devez configurer Easycom pour utiliser le serveur Easycom approprié.

Pour cela, ajouter le numéro de port à la fin de l'adresse IP, séparé par " : ".

Exemple :

SYSTEMAS**:6078**

192.168.0.10**:6078**

[2001:db8::1428:57ab]:**6078**

Cette valeur peut être spécifiée au niveau de l'outil "Easycom configuration", ou bien dans les propriétés de connexion de l'application.

Licence d'utilisation

Licences

L'utilisation complète d'Easycom implique l'acquisition de la licence appropriée.

Vous devez enregister votre numéro de série produit « Easycom for Windev » afin d'obtenir votre clé d'activation qui doit être saisie.

Clé temporaire

Lorsque vous téléchargez le produit à des fins d'évaluation, une clé temporaire est calculée avec une date limite.

Contactez le service commercial pour plus d'informations sur les clés temporaires.

Développement et déploiement

Pour utiliser Easycom dans l'environnement de développement (WinDev ou WebDev), maintenir des applications et les déboguer une licence Développement est nécessaire.

La licence Déploiement permet uniquement aux applications de se connecter à l'AS400. Le constructeur de DDS (export de fichiers) et le constructeur RPC-DTAQ ainsi que WDMAP et WDETAT peuvent être utilisés avec une licence Déploiement.

Nombre de connexions simultanées

Le nombre de connexions autorisées est fixé par la clé d'activation, il détermine le nombre de jobs et d'instances d'applications qui peuvent se connecter simultanément.

Plusieurs instances d'une application sur un même poste vont donc utiliser plusieurs connexions, de même que plusieurs postes avec la même application.

Attention !!! Plusieurs connexions dans un même projet vont également ouvrir plusieurs jobs et consommer plusieurs connexions.

Déploiement illimité

C'est la licence qui permet un nombre de connexions illimité, recommandée particulièrement pour du déploiement WebDev où il est difficile de maîtriser le nombre de connexions.

Licence par AS/400

Pour accéder à plusieurs AS/400, vous devez disposer d'**une licence par AS/400**.

Changement d'AS/400

En cas de changement d'AS/400, remplir et renvoyer le formulaire.

Enregistrement numéro de série produit

Vous devez obligatoirement enregistrer le numéro de série produit acheté, afin de l'affecter à un AS400 (numéro de série et partition). Vous recevrez alors votre licence à enregistrer sur l'AS400. Vous bénéficierez alors de la garantie produit d'un mois, pendant laquelle vous pourrez télécharger le produit sur notre site web, et bénéficier du support technique.

Au-delà de la période de garatie, il vous faudra soucrire à un contrat de maintenance/assistance pour continuer à bénéficier du support technique et des mises à jour des produits. Contactez le service commercial si nécessaire.

Enregistrement de la licence sur l'AS400

La clé d'activation est obtenue après avoir enregistré le numéro de série produit sur notre site web. Vous recevrez un email du service commercial contenant la clé à enregistrer sur la machine IBM i, avec la commande EASYREG :

```
License  .□. . . . . .   WEBDEV17

Special Development License  . .   *YES

Company name . . . . . .

Activation Key . . . . . .

Number of connections  . . . . .   1

Authorized partition ID (1->n)     1

EASYCOM'S Version  . . . . . .   3

EASYCOM'S Options  . . . . . .   0

Expiration date (ddmmyyyy) . . .   00000000

    Authorized Proc. group . . . . .   *

Extended license . . . . . .


(Cette clé d'activation a été générée pour l'AS/400 n°        )
```

Pour enregistrer la clé, il faut utiliser un terminal AS/400, et exécuter les commandes suivantes :

ADDLIBLE EASYCOM

EASYREG avec les infos contenues dans l'email.

Voir : Clé d'activation

Développement

Gestion des connexions

Connexions

Une connexion peut être définie dans l'analyse ou par programmation (variable de type Connexion ou description par la fonction `HDécritConnexion`).

L'ouverture d'un fichier va automatiquement ouvrir la connexion associée. Elle peut être ouverte explicitement avec `HOuvreConnexion`, notamment pour utiliser les fonctions spécifiques.

Il est en fait conseillé de toujours utiliser HOuvreConnexion dans un programme, même si les paramètres de l'analyse sont suffisants. Cela permet de maîtriser à quel moment ouvrir et fermer la connexion. Autrement la connexion est fermée dès que tous les fichiers l'utilisant le sont.

Les mots de passe longs (V5R2) sont supportés.

Une connexion correspond à un JOB actif dans le sous-système Easycom. Attention en cas d'un nombre de licences limité. Chaque job correspond à l'utilisation d'une licence.

Il est possible d'avoir plusieurs connexions dans une même analyse et de combiner ainsi différents accès : gérer plusieurs AS400 et/ou profils ou bien des fichiers AS400 et Hyper File... ou d'autres bases externes.

Remarque : un fichier AS400 peut être ouvert directement avec la fonction HDeclareExterne.

Les paramètres de la connexion

Tous les paramètres sont optionnels, on peut tout à fait utiliser la syntaxe suivante :

```
HOuvreConnexion(NouvelleConnexion,"","","","",hAccèsNatifAS400)
```

et compléter ensuite la fenêtre de login, à condition que les boites de dialogues ne soient pas désactivées (voir aussi automatiser la connexion).

Le provider est bien sûr *`hAccèsNatifAS400`*.

Le mode d'accès peut être *`hOLectureEcriture`* ou *`hOLecture`*.

Les trois paramètres essentiels sont :

- L'adresse IP de l'AS/400 ou son nom (à condition qu'il figure dans le fichier host ou dans le DNS)
- Le nom de l'utilisateur (profil)
- Son mot de passe

Les **informations étendues** peuvent être utiles pour spécifier les paramètres propres à une connexion AS/400, comme le nom du travail par exemple.

Variable de type Connexion

La variable de type connexion permet de définir l'ensemble des propriétés d'une connexion, elle est ensuite passée comme paramètre à la fonction `HOuvreConnexion`.

Exemple

```
MaConnexion est une Connexion
// Description de la connexion
```

```
MaConnexion..Utilisateur = "USER"
MaConnexion..MotDePasse = "PASSWORD"
MaConnexion..Serveur = "NOM_ou_ADRESSE_IP"
MaConnexion..BaseDeDonnées = ""
MaConnexion..Provider = hAccèsNatifAS400
MaConnexion..Accès = hOLectureEcriture
MaConnexion..InfosEtendues =
"<EASYCOM>"+CRLF+"JOBNAME=+CRLF+"</EASYCOM>"

HOuvreConnexion(MaConnexion)
```

Les infos étendues doivent respecter les sauts de ligne.

Il est possible de ne pas définir les paramètres ou plus précisément de laisser des chaînes vides. Dans ce cas, si les boites de message Easycom ne sont pas désactivées, apparaît l'écran de login.

Connexion dans l'analyse

Vous pouvez définir une ou plusieurs connexions dans une analyse et leur associer des fichiers.

Au moment de l'importation de fichiers dans une analyse, la connexion correspondante est automatiquement créée et les fichiers importés sont associés par défaut à cette analyse.

Il est bien sûr possible d'ajouter des connexions dans l'analyse et de leur associer les fichiers.

L'ouverture d'un fichier de l'analyse va automatiquement utiliser la connexion associée. Si tous les paramètres sont renseignés la connexion est donc transparente, dans le cas contraire la boite de dialogue de connexion apparaît.

HOuvreConnexion

Une connexion s'ouvre avec la fonction HOuvreConnexion.

Il peut s'agir d'une connexion définie dans l'analyse, d'une connexion décrite par la fonction HDecritConnexion ou d'une variable de type Connexion.

```
HOuvreConnexion("MaConnexion","QPGMR","QPGMR","194.215.168.102",
hAccèsNatifAS400,hOLectureEcriture,"")
```

En single sign-on de type EIM, utilisez la valeur spéciale « *KERBAUTH » comme nom d'utilisateur, sans modifier le mot de passe. Voir CFGEACEIM pour la configuration côté IBM i.

En Accès Natif AS/400, la connexion à la base de données revient à se connecter à l'AS/400 et donc à créer une session et un nouveau job.

HDécritConnexion

Cette fonction permet de créer une connexion qui n'est pas définie dans l'analyse. Il faudra ensuite associer les fichiers à cette nouvelle connexion avec la fonction HChangeConnexion.

```
bResultat = HDécritConnexion(Connexion , Utilisateur, , Passe,"" ,
AdresseIP, hAccèsNatifAS400 [,Accès [, Infos_étendues [, Options]]])
```

On retrouve les mêmes paramètres que la variable de type Connexion.

HChangeConnexion

HChangeConnexion permet de changer la connexion associée à un fichier. La nouvelle connexion doit être connue et définie lors de ce changement.
Cette nouvelle connexion sera utilisée lors de la prochaine ouverture du fichier.

```
HChangeConnexion(Fichier, Connexion)
```

Exemple

Un fichier CLIENTS qui existe sur l'AS400 et au format Hyper File, dans l'analyse ce fichier est associé à la connexion Hyperfile. La fonction HChangeConnexion permet de travailler avec le même fichier à la fois en Hyper File et en fichier AS/400.

Attention, puisque le fichier n'a pas d'infos étendues pour l'associer à un fichier AS/400 (MAIN) il faut utiliser l'option IGNORE_EMPTY_EXTINFO dans les infos étendues de la connexion et le fichier sera recherché dans la LIBL du profil. Il doit évidemment avoir une structure cohérente sur l'AS/400 avec sa description dans l'analyse.

```
// ouverture par défaut : fichier HF
HOuvre(Clients)
......... // traitements sur le fichier HyperFile

// Décrire et ouvrir la nouvelle connexion sur l'AS/400
HOuvreConnexion("MaConnexion", "QPGMR", "QPGMR", " 194.206.160.105
", hAccèsNatifAS400, hOLectureEcriture,
"<EASYCOM>"+CRLF+"IGNORE_EMPTY_EXTINFO=+CRLF+"</EASYCOM>")


HChangeConnexion(Clients,MaConnexion)
HOuvre(Clients)
...
HFerme(Clients)
HFermeConnexion(MaConnexion)
```

La syntaxe HChangeConnexion("*", NouvelleConnexion) va changer de connexion pour tous les fichiers.

HFermeConnexion

La fonction HFermeConnexion va explicitement fermer tous les fichiers ouverts de la connexion et fermer la connexion elle-même, et donc arrêter le job associé.

Cette fonction n'est pas obligatoire, la fermeture de l'application ou du dernier fichier ouvert vont également fermer la connexion et arrêter le job.

ASPropriete

Cette fonction donne permet de définir différentes propriétés sur un fichier ou une source de donnée :

- Activation et chemin du fichier d'alias (uniquement pour un fichier),
- Sélection du membre (uniquement pour un fichier).
- Options de curseur et de cache (fichier ou source de donnée),

Cette fonction permet également de continuer à travailler avec les fichiers d'alias, lors d'une migration d'un projet WinDev 5.5

Syntaxe

```
Result = ASPropriete(NomFichier, Propriété, Valeur [, Connexion])
```

En anglais : ASProperty

Paramètres

Result

Booléen. Vrai si la propriété a été affectée, Faux en cas d'erreur.

NomFichier

Chaîne.

Nom du fichier de l'analyse pour lequel on souhaite modifier une propriété ou chaîne vide pour l'ensemble des fichiers.

Propriété

Chaîne.

Une des valeurs du tableau ci-dessous :

Propriétés	*Sous-Type*	*Valeurs*
MEMBER	chaîne	Nom du membre du fichier à utiliser. Le fichier doit être ouvert après le ASPropriete. Le membre peut également être spécifié dans l'option MAIN des infos étendues du fichier : `MAIN=BIBLIO/FILE(MEMBER)`
ALIASPATH	chaîne	Chemin où se trouve le fichier d'alias : nom_fichier._as
ONLYALIAS	booléen	Ignore les infos étendues pour n'utiliser que les informations du fichier d'alias.
CACHESIZE	entier	Nombre d'enregistrement maximum dans le cache.
FORWARDONLY	booléen	Curseur le plus rapide et le moins gourmand en ressource : le fichier est lu du début vers la fin.
CACHEDINSERT	entier	Cache en écriture : un `HAjoute` ne va pas directement écrire l'enregistrement mais alimenter un tampon avec le nombre d'enregistrements spécifié. Cette option doit être utilisée avec la plus grande prudence.

Valeur

Selon la propriété et le type du tableau.

Connexion (optionnel)

Connexion - Nom de la connexion

Fichier d'alias

Le fichier d'alias était utilisé en WinDev 5.5 pour identifier les fichiers AS/400. C'est un fichier texte qui se trouve dans le répertoire où devrait se trouver son équivalent HyperFile.

```
[NOM_DU_FICHIER]
$FILE=BIBLIO/FICHIER ou FICHIER
CLE1=BIBLIO/LOGIQUE1
CLE2=BIBLIO/LOGIQUE2
...
$JNAL=TRUE // si le fichier est journalisé
$READONLY=TRUE // ouverture en lecture seule
```

$FILE correspond désormais au MAIN des infos étendues du fichier.

Les logiques associées aux clés sont dans les infos étendues des rubriques (LFSYSNAME).

$JNAL correspond à l'option JOURNALED des infos étendues du fichier.

$READONLY correspond au mode d'ouverture de la connexion (HOuvreConnexion).

Remarques :

1. Si un fichier d'alias est utilisé, la modification de la LIBL du JOB ne sera pas prise en compte (ADDLIBLE).
2. Pour utiliser un fichier d'alias avec HExécuteRequêteSQL, il faut utiliser l'option hRequêteDéfaut

```
HExécuteRequêteSQL(marequete,MaConnexionpower8,hRequêteDéfaut,schaine)
```

Exemple de fichier d'alias : SP_CUST._as

[SP_CUST]
$FILE=EASYCOMXM2/SP_CUST
CUST_ID=EASYCOMXM2/SP_CUST_CU
FIRSTNAME=EASYCOMXM2/SP_CUST_NA
STATE=EASYCOMXM2/SP_CUST_ST

Exemples

Utiliser les fichiers d'alias

```
Résultat est un booléen

// Indique dans quel répertoire trouver tous les fichiers d'alias
Résultat=ASPropriete("", "ALIASPATH", "C:\Program Files\Projet")

// ignorer les infos étendues, avant l'ouverture du fichier
Résultat=ASPropriete("", "ONLYALIAS", "VRAI")
```

Cache et curseur

'Doper' les performances de lecture "brute" de données provenant d'un HExecuteRequeteSQL.

```
// Il faut exécuter la requête avant d'utiliser ASPROPRIETE
HExécuteRequête(MaRequete)
//Cache de 1000 enregistrements
ASPropriete(MaRequete, "CACHESIZE", "1000")
ASPropriete(MaRequete, "FORWARDONLY", "1")
```

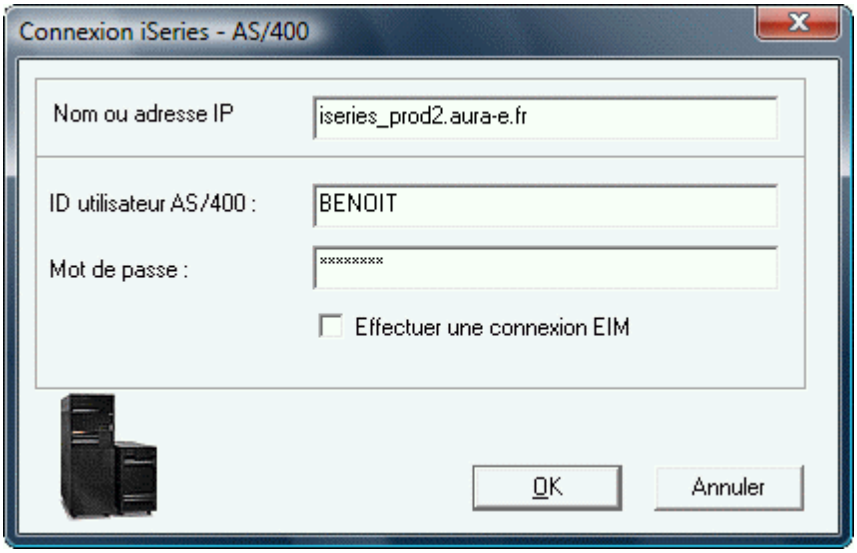
Le gain de performance peut théoriquement atteindre 50% sur la partie "lecture" du résultat.

Connexion automatique

Les propriétés de la connexion peuvent être définies par programmation, permettant une boîte de login personalisée ou bien de cacher l'authentification dans le programme.

Si le profil et le mot de passe se trouvent dans la connexion définie dans l'analyse ou bien sont passées en paramètre dans la fonction HOuvreConnexion, la connexion s'établit directement.

Dans le cas contraire, si les boîtes de message EASYCOM ne sont pas désactivées, apparaît la fenêtre de login Easycom demandant le nom du profil et le mot de passe.



Il existe deux solutions pour éviter la saisie systématique du login/mot de passe :

1) easycom.ini (pour les développeurs, sécurité faible) :

En renseignant l'utilisateur et le mot de passe dans le fichier Easycom.ini, qui se trouve dans le répertoire de Windows (ou dans le répertoire de l'application).

Dans la section `[GENERAL]`, saisir les entrées suivantes (par exemple) en majuscules :

```
[GENERAL]
Uid=QPGMR
Pwd=QPGMR
```

Attention, les valeurs sont alors écrites en clair dans le fichier, et aucun cryptage n'est opéré. Cette solution peut être commode et confortable dans un environnement de développement mais n'est pas conseillée en déploiement.

2) Single Sign On

La connexion peut également être automatisée en « mémorisant » la première signature (Sign On).

Cette fonctionnalité fonctionne uniquement si la gestion du Single Sign On est activée côté client (voir EASYCOM Configuration - Single Sign On) et côté serveur.

Le type de single sign-on recommandé est le type "EIM", géré par l'OS/400.

Infos étendues

Connexion

Propriétés au niveau de la connexion

Elles peuvent également être passées comme paramètre (chaîne) dans les fonctions HOuvreConnexion ou dans une variable de type Connexion. Dans ce cas, chaque option est séparée par un CRLF ou RC (saut de ligne)

Par exemple :

```
sInfosEtendues est chaîne
sIpAddr est chaîne


sIpAddr = "power8.aura.fr"
sInfosEtendues=[
```

```
 <easycom>
 JobName=test
 InitLibl=CR;PROD
 ssl=vrai
 </easycom>
]


SI PAS HOuvreConnexion(MaConnexion,"*kerbauth",
"",sIpAddr,hAccèsNatifAS400,hOLectureEcriture,sInfosEtendues)
ALORS
 Erreur("Erreur de connexion: "+HErreurInfo())
 FinProgramme()
FIN
```

AUTOJOURNAL

Valeur : Vrai (par défaut), Faux

En importation de DDS, récupère et ajoute dans les infos étendues du fichier la propriété JOURNALED=TRUE" pour les fichiers journalisés.

CODEPAGEFILE

Chemin vers un fichier .cpg sur le disque, côté PC qui contient une page de codes de conversion ASCII/EBCDIC.

En général cette option n'est utile que pour des pays particuliers (Chine, Corée, Japon) ou des configurations inhabituelles.

Les fichiers de conversion (.cpg) ne sont pas installés par défaut.

Voir : Caractères spéciaux.

```
CODEPAGEFILE=C:\Easycom\eansi297.cpg
```

CONNECTION TIMEOUT et COMMAND TIMEOUT

Attention, si une erreur de timeout "command" survient, il faut fermer pour rouvrir la connexion.

0 signifie attente infinie.

DATETYPE / TIMETYPE

Il est possible d'associer des rubriques de type Date ou Heure sous l'analyse à des rubriques de formats différents sur l'AS/400.

Dans ce cas cette option peut être utile en mode SQL avec le mode

HRequêteDéfaut : si une colonne de type date ou heure associée à un autre type de donnée est utilisée dans une clause « Where », la valeur est en "dur" et au format WinDev, l'option indique à Easycom comment reformater cette valeur.

Cette option peut être définie pour tous les fichiers (infos étendues de la connexion) ou pour un fichier en particulier (infos étendues du fichier), elle s'applique alors par défaut à toutes les rubriques de type Date ou Heure.

Elle peut également être définie au niveau de la rubrique par l'option NATIVETYPE. Lorsque des fichiers Hyper File sont exportés puis réimportés (ou synchronisés) et que le type pour des rubriques date ou heure a été modifié, cette option est automatiquement récupérée au niveau de la rubrique.

Par exemple :

- un champ date associé à un CHAR (texte) devient WHERE DATE='20041202'
- un champ date associé à un PACKED (numérique) devient WHERE DATE=20041202 (sans les quotes)

Même principe pour les heures et horodatages.

Les valeurs possibles sont :

0 - caractère

1 - entier sur 2 octets

2 - entier sur 4

3 - flotant simple précision (4 octets)

4 - flotant double précision (8 octets)

6 - condensé

7 - étendu

8 - date

9 - heure

10 - horodatage

13 - entier sur 8

Pour les dates, le format doit toujours contenir 8 positions disponibles (numérique ou caractère) et être de type "aaaammjj".

Pour les heures, le format doit toujours contenir 6 positions disponibles (numérique ou caractère) et être de type "hhmmss".

Exemple :

```
DATETYPE=0
```

Tous les champs de type date sont associés à des char (sauf si la rubrique contient un NATIVETYPE différent)

DRVOPTIMISTIC

Active le mode de verrouillage automatique, notamment pour WebDev.

Dans ce mode, toute mise à jour (HModifie) va entraîner un blocage automatique avec vérification.

Avec WebDev cette option évite une erreur de blocage si une gestion d'erreur (HSurErreur("*", *hErrBlocage*) n'est pas faite.

DUPPATH

"Duplicated Path" ou chemin dupliqué.

Possibilité de faire un HFiltre avec condition sur une même clé dans deux contextes Hyper File différents en cochant l'option 'Contexte HF indépendant'.

Si l'option n'est pas active, l'historique des messages fera apparaître une erreur ("Ouverture partagée non admise pour la requête.") et le résultat sera vide.

EIM_LOOKUP_INFO

L'information "Eim Lookup" est une chaine de caractères utilisée en mode EIM. Le mode EIM peut être utilisé lors d'une connexion par certificat (en SSL) ou par une authentification Kerberos.

Cette information est utilisée pour résoudre les ambigüités de correspondance d'utilisateur OS/400. Par exemple une authentification peut induire un utilisateur OS/400 ou un autre en fonction de cette information 'Lookup Info'.

EXTRAIDX

Cette option concerne l'utilisation des loupes sur les clés composées.

Par défaut `EXTRAIDX=1`

FIELDNAMES_MUST_MATCH

Il s'agit d'une option pour compatibilité avec les anciennes versions.

Détermine si les noms de champs AS/400 doivent correspondre ou non aux noms de champs de l'analyse (que le fichier soit en mode HF ou en mode AS/400)

Valeur par défaut : vrai

Si les noms de champs ne correspondent pas mais que leur structure correspond, vous pouvez utiliser cette option pour éviter le message d'erreur à l'ouverture du fichier (attention dans ce cas à bien vérifier l'ordre physique des champs dans l'analyse)

Pour activer le contrôle par ordre physique (donc sans tenir compte des noms de champs), utiliser :

FIELDNAMES_MUST_MATCH=Faux

FORCELIBL

Permet d'ignorer les bibliothèques présentes dans l'analyse : travail, en LIBL complet.

Lorsque cette option est égale à 1, les informations de bibliothèque pouvant être présentes dans l'analyse sont ignorées.

Par exemple, si des infos étendues contiennent :

PF=LIB1/FICHIER

Le fichier « FICHIER » sera cherché via la LIBL courante et non dans LIB1. Si LIB1 n'est pas dans la LIBL le fichier ne sera pas ouvert.

HFMAXKEY

Cette option concerne l'utilisation de séquences de tri particulières dans lesquelles le dernier caractère ne correspond pas au code FF.

Par défaut, *hValMin* est équivalent à Caract(0) et *hValMax* est équivalent à Caract(255), mais si le dernier caractère a un autre code, elle sert à spécifier au moteur Hyperfile que le code maximal pour les filtres et recherches sur une clé composée réduite est différent.

```
HFMAXKEY=TRUE ou FALSE
```

IGNORE_EMPTY_EXTINFO

Ignore les "Infos étendues".

Dans certains cas un fichier peut ne pas contenir d'infos étendues : par exemple un fichier Hyper File dont on change la connexion ou un fichier dont le type est modifié (Hyper File vers AS/400) sans synchronisation avec la base externe.

Dans ce cas, en l'absence d'infos étendues et donc de liens vers le fichier physique (option MAIN du fichier) ou vers les fichiers logiques (option LF des clés), Easycom va chercher un fichier logique qui a le même nom que la clé dans la LIBL du job.

IMPORT_DFT

Importe les valeurs par défaut des champs dans l'analyse.

Valeur par défaut : vrai

IMPORT_SELOMIT

Permet d'importer volontairement les fichiers ayant des conditions de sélection / omission en tant que clé.

Attention aux risques par rapport aux tables avec loupes et au changement de clé (HchangeClé).

INITLIBL

Liste des bibliothèques initiales séparées par des " ; " ajoutées à la liste de l'utilisateur.

Voir aussi les bibliothèques initiales d'un job Easycom.

```
INITLIBL=PROD2005;STATS2005
```

Il est également possible de définir la liste initiale des bibliothèques pour toutes les connexions dans le fichier easycom.ini :

```
[WINDEV]
INITLIBL=PROD2005;STATS2005
```

Remarque : les options présentes dans easycom.ini sont utilisées uniquement si aucune option INITLIBL n'est précisée dans infos étendues de connexion.

LITERALCASE

Gestion de la casse (majuscules / minuscules) à l'importation des fichiers AS/400.

Depuis la version 9, les noms de fichiers sont importés par défaut en minuscules, si vous souhaitez contrôler le format du nom des fichiers et des rubriques, utilisez cette fonction avec les paramètres ci-dessous :

A	MAJUSCULE
a	Minuscule
#nx	Prochaine action x pour les n caractères
*x	Action x sur toute la chaîne depuis la position en cours
[	Se positionne à la fin, et change de sens (négatif)
]	Se positionne au début, et change de sens (positif)
<	Change de sens (négatif)
>	Change de sens (positif)

Exemples d'expression :

<aucune>	Tout en minuscules
A*a	Première lettre majuscule, toutes les autres en minuscules (défaut)
#3A#3a*A:	3 premières lettres majuscules, 3 suivantes minuscules, la suite en majuscules
#3A[#3A:	3 premières et 3 dernières en majuscules, le milieu en minuscules

L'option doit bien sûr être présente dans les infos étendues de la connexion avant l'importation des fichiers.

Exemple : Importer les fichiers avec les Noms en Majuscules

```
<EASYCOM>
LITERALCASE=*A
</EASYCOM>
```

JOBNAME

Nom du Job sur l'AS/400, par défaut c'est le nom du PC.

Le nom du job peut également être spécifié côté AS400 par les programmes de sécurité ou les propriétés d'EACJOBD.

```
JOBNAME=WINDEV
```

LEADINGSPACES

Valeurs : 0 (défaut) ou 1

Complète par des espaces la valeur du champ clé (compatibilité pour WinDev 55).

Dans les propriétés d'un fichier au niveau de l'analyse l'option "Mode compatible 5.5" propose de compléter les chaînes par des espaces (au lieu du caractère binaire zéro à la fin). Dans ce cas il faut utiliser la fonction Complete dans les filtres et les recherches. Cette option est l'équivalent Easycom.

Par défaut la valeur est égale à 0.

```
LEADINGSPACES=1
```

ONLYSHORTFIELDNAMES

Valeurs : 0 (par défaut) ou 1

Pour utiliser uniquement les noms courts de champs.

```
ONLYSHORTFIELDNAMES=1
```

PGMNAME

Cette propriété permet de fixer le programme Easycom à appeler lors de la connexion. Par défaut il s'agit du programme « EASYCOM » dans la bibliothèque dans laquelle s'exécute le programme EASYCOMD utilisé lors de la connexion, soit « EASYCOM » par défaut.

SQLNAMING

SYS ou SQL (SYS par défaut)

Permet d'utiliser le "." (point - SQL) à la place du "/" (slash - SYS) dans les requêtes SQL.

Attention, dans la syntaxe par "." (SQL), la liste des bibliothèques ne peut être utilisée et il faut préciser explicitement le nom de la bibliothèque dans la requête.

```
SQLNAMING=SQL
```

SHOWDIALOGS

Permet de désactiver les boîtes de dialogues d'Easycom.

On peut également désactiver les boites de dialogues depuis Easycom Configuration.

```
SHOWDIALOGS=0
```

SSL

Permet de choisir le mode d'utilisation de SSL.

Les valeurs possibles sont :

- **Vrai** : La connexion sera tentée en SSL
- **Faux** : ne pas utiliser SSL pour la connexion. Permet de forcer une connexion normale, même si une connexion SSL est configurée via Easycom configuration.
- **Obligatoire** : La connexion doit se faire en SSL, sinon échec de la connexion

Voir aussi :

SSL

Configuration SSL cliente

SSL_INTF

Permet de choisir l'interface SSL cliente à utiliser. L'interface Windows est conseillée.

Les valeurs possibles sont :

- Windows (default) : l'interface utilisée est Windows. Le magasin de certificats utilisé sera celui de Windows, accessible via certmgr.msc
- OpenSSL. Utiliser l'interface OpenSSL.

Voir aussi :

SSL

Configuration SSL cliente

SSL_CAFILE

Valeur du fichier de certificat à utiliser. N'est utile que si l'interface est OpenSSL.

Voir aussi :

SSL

Configuration SSL cliente

SSL_CAPATH

Valeur du chemin de certificats à utiliser. N'est utile que si l'interface est OpenSSL.

Voir aussi :

SSL

Configuration SSL cliente

STRICTIDENTICAL

Cette option concerne l'utilisation de recherche à l'identique.

Cette option permet de faire une « vraie » recherche à l'identique, et en cas d'échec la position courante dans le fichier ne change pas.

```
STRICTIDENTICAL=1 (Par défaut : 0)
```

SQLFULLPRECISION

Valeurs : 0 (par défaut) ou 1

Gestion des nombres longs dans les requêtes sur l'AS/400.

Le format monétaire de WinDev peut prendre une valeur entre -604 462 909 807 314 587,353 087 et +604 462 909 807 314 587,353 087. Dans le cas ou des valeurs dépassent ces limites (plus de 17 chiffres significatifs pour la partie entière, 6 pour la partie décimale), cette option propose une conversion automatique dans un format texte qui conserve la précision.

`SQLFULLPRECISION=1` : transformation en texte

`SQLFULLPRECISION=0` : transformation en monétaire

SQLHPOS

Amélioration sensible des comportements de l'objet table en mode « fichier », concernant les barres de défilement.

Le fonctionnement de l'objet table en mode « fichier », avec ascenseur proportionnel demande des traitements particuliers, par exemple déterminer le nombre d'enregistrements (en tenant compte des filtres), savoir aller à un pourcentage de l'ensemble ou encore déterminer la position après avoir effectué une rechercher par clé. Toutes ces actions nécessitent des actions serveur qui peuvent être consommatrices de ressources sur l'AS/400.

L'option SQLHPOS=1 permet de choisir si l'on veut ou non utiliser ces traitements particuliers afin de présenter des tables en mode « fichier » de comportement plus intuitif.

Valeur par défaut : 0

Avec SQLHPOS=0

- HLitRecherche suivi de TableAffiche va positionner la scrollbar au milieu de l'ensemble (mais les données affichées sont bien entendu les bonnes)
- Un positionnement en pourcentage peut entraîner un nombre important de lectures séquentielles

Avec SQLHPOS=1

- HLitRecherche suivi de TableAffiche va positionner la scrollbar extactement à l'endroit correspondant. Mais des traitements supplémentaires sont demandés à l'AS/400 (sous forme SQL).
- Un positonnement en pourcentage est plus rapide pour un nombre important d'enregistrements. Par contre cela utilise des requêtes SQL si la table est basée sur un fichier.

SQLIDX

Ajout des loupes dans les tables fichiers reliées aux requêtes.

Les loupes correspondent aux champs indexés sur lesquels on peut faire un tri ou une recherche, lorsque cette option est activée, des index sont crées pour chaque colonne de la requête, cette option a évidemment des répercussions sur les performances et doit être utilisée avec prudence.

```
SQLIDX=1
```

SQLLIVE

Permet d'activer la possibilité d'effectuer des modifications d'enregistrements sur des requêtes.

Par défaut SQLLIVE vaut 0 : aucune modification n'est possible dans une requête.

Lorsque SQLLIVE vaut 1 : seuls les fichiers avec l'option SQLUPDATABLE=1 (infos étendues du fichier) sont modifiables.

Lorsque SQLLIVE vaut 3 : tous les fichiers sont modifiables à moins de contenir l'option SQLUPDATABLE=0 dans les infos étendues du fichier.

Il est également possible d'exclure la modification pour certaines rubriques avec l'option SQLUPDATABLE=0 dans les infos étendues de la rubrique.

Remarque : Le SELECT ne doit pas contenir '*', il faut renseigner explicitement le nom des rubriques à modifier.

Le SELECT doit correspondre à des enregistrements cohérents (pas de GROUP BY) mais peut concerner plusieurs fichiers.

```
SQLLIVE=3
```

TCP_VERSION

Cette option permet de forcer la version de TCP/IP à utiliser. Par défaut, la version est déterminée par la requête DNS. Les valeurs possibles sont « 4 » pour IPv4 ou « 6 » pour IPv6.

TCPIP_FATAL

Toute erreur TCP/IP est fatale si l'option est activée. Elle est à 1 par défaut depuis la version 12.

Cela permet de gérer les erreurs TCP/IP par une procédure d'exception générale (et par exemple relancer l'application à une fenêtre donnée en cas d'erreur d'un certain type et en fonction du contexte).

UNLOCK

Déverrouillage d'Easycom

Ce paramètre fonctionne conjointement avec l'activation du script de contrôle d'accès sur la partie serveur Easycom (voir les options de sécurité avancée côté AS400).

Il contient le mot de passe qui sera vérifié par le programme EACP003 .

```
UNLOCK=password
```

USER_CERT_FILE / USER_CERT_PKEY_FILE / USER_CERT_PASSPHRASE

Ces propriétés ne sont valables qu'en mode OpenSSL.

Elles permettent de choisir le certificat client à utiliser pour la connexion. En mode Windows, le certificat est choisi automatiquement à partir du magasin de certificats de Windows.

USER_CERT_FILE désigne le chemin du certificat client.

USER_CERT_PFILE_FILE désigne le chemin du fichier contenant la clé privée. Ce fichier doit se trouver dans un emplacement protégé.

USER_CERT_PASSPHRASE est égal au mot de passer permettant de décrypter la clé privée, si elle a été cryptée.

Remarque : Ces propriétés utilisent des fichiers aux formats reconnus par OpenSSL.

Fichiers

Propriétés au niveau fichier

MAIN

Définition de la source du fichier de l'AS/400.

```
MAIN=Nom de la Bibliothèque/Nom du Fichier Physique
```

ou

```
MAIN=Nom du fichier
```

Le nom peut être complété de certaines options pour gérer des types de fichiers particuliers :

Pour spécifier un membre :

```
MAIN=BIBLIO/FILE(MEMBER) ou MAIN=FILE(MEMBER)
```
(voir aussi ASPropriete)

Pour spécifier un format :

```
MAIN=BIBLIO/FILE *RFORMAT=NOM_DU_FORMAT
```
(voir aussi fichiers multi-formats)

Ce nom peut également être une requête SQL, les rubriques doivent évidemment respecter la structure du fichier

```
MAIN=SELECT * FROM EASYCOM/SP_CUST ORDER BY CUST_ID
```

JOURNALED

Indique que le fichier est journalisé et qu'il peut être utilisé dans une transaction.

En importation cette information est automatiquement récupérée, pour l'ignorer on peut utiliser l'option AUTOJOURNAL des infos étendues de la connexion.

```
JOURNALED=TRUE
```

PFSYSNAME

Nom court (système) du fichier physique.

Il est possible de n'utiliser que les noms courts en mettant à 1 la propriété ONLYSHORTFIELDNAMES des infos étendues de la connexion.

SQLUPDATABLE

Pour pouvoir modifier le résultat d'une requête SQL, les infos étendues de la connexion doivent contenir l'option SQLLIVE avec une valeur de 1 ou 3.

Si SQLLIVE=1, les fichiers qui peuvent être modifiés doivent contenir l'option SQLUPDATABLE=1.

Si SQLLIVE=3, tous les fichiers sont modifiables par défaut mais il est possible d'exclure un fichier particulier (empêcher toute mise à jour depuis une requête), avec l'option SQLUPDATABLE à 0.

DATETYPE / TIMETYPE

Cette option associe tous les champs date et heure du fichier à un autre type de format pour un formatage automatique de requête en mode *HRequêteDéfaut*. Elle peut également être définie au niveau de la connexion (voir détail) ou au niveau de la rubrique par l'option NATIVETYPE (voir ci-après).

Rubriques

Propriétés au niveau rubrique

LF

Chemin d'accès du fichier logique correspondant à une clé.

```
LF=Nom de la Bibliothèque/Nom court du Fichier Logique
```

SYSNAME

Nom court (système) pour une rubrique non-clé,s clé unique et clé avec doublon

```
SYSNAME=Nom court de la rubrique
```
(10 caractères maxi)

LFSYSNAME

Nom court (système) pour le fichier logique associé à une rubrique clé composée

```
LFSYSNAME=Nom court du Fichier Logique
```
(10 caractères maxi)

SQLUPDATABLE

Pour pouvoir modifier le résultat d'une requête SQL, les infos étendues de la connexion doivent contenir l'option SQLLIVE avec une valeur de 1 ou 3.

Si SQLLIVE=1, les fichiers qui peuvent être modifiés doivent contenir l'option SQLUPDATABLE=1

Si SQLLIVE=3, tous les fichiers sont modifiables à moins de contenir SQLUPDATABLE=0

Dans ces deux cas, toutes les rubriques sont modifiables par défaut mais il est possible d'exclure certaines rubriques avec l'option SQLUPDATABLE à 0 dans les infos étendues de la rubrique.

NATIVETYPE

Cette option est automatiquement insérée à l'importation lorsqu'un champ de type Date ou Heure a été associé à un autre format. Voir les valeurs possibles dans l'option DATETYPE/TIMETYPE des infos étendues de la connexion.

Par exemple si un champ de type Date est associé à une zone CHAR (sur 8 caractères) on aura :

```
NATIVETYPE=0
```

Accès aux données de l'AS/400

Importer des fichiers AS400 dans l'analyse Windev

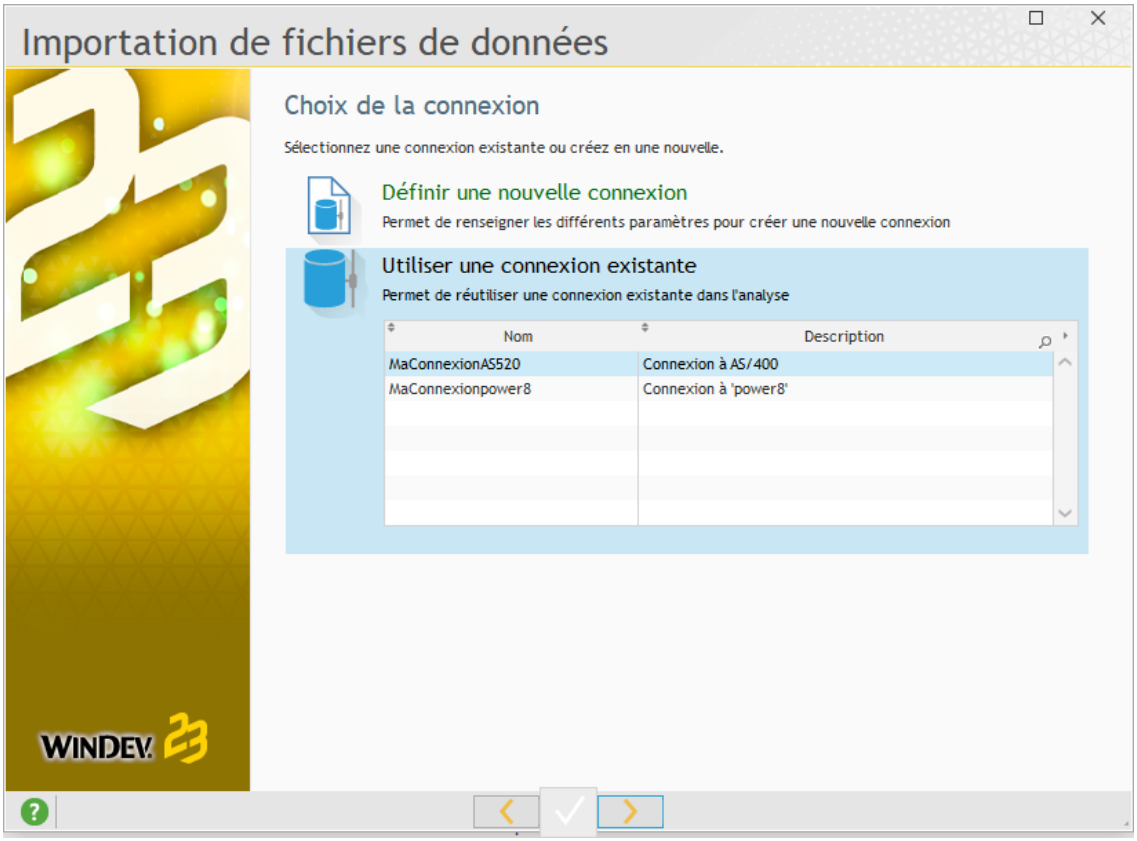
Import des descriptions de fichiers

L'importation des descriptions de fichiers AS400 se fait très simplement depuis l'analyse Windev :

> *Sélectionner AS/400 comme source de données, puis l'option "Utiliser des fichiers d'une base existante" et "Accéder aux données dans leur format actuel".*

Si vous souhaitez simplement transférer des données dans un fichier Hyperfile, voir Import de données.

L'importation de fichiers va créer une connexion mais si vous souhaitez utiliser un certain nombre d'options (contrôle de la casse des noms de fichiers et rubriques, formats de date...) il est nécessaire de les avoir renseignées avant l'importation : voir les infos étendues de la connexion.



Mode : "Accès natif AS/400",

Serveur : adresse IP ou nom de l'AS400

Nom d'utilisateur : nom du profil

Mot de passe : mot de passe du profil

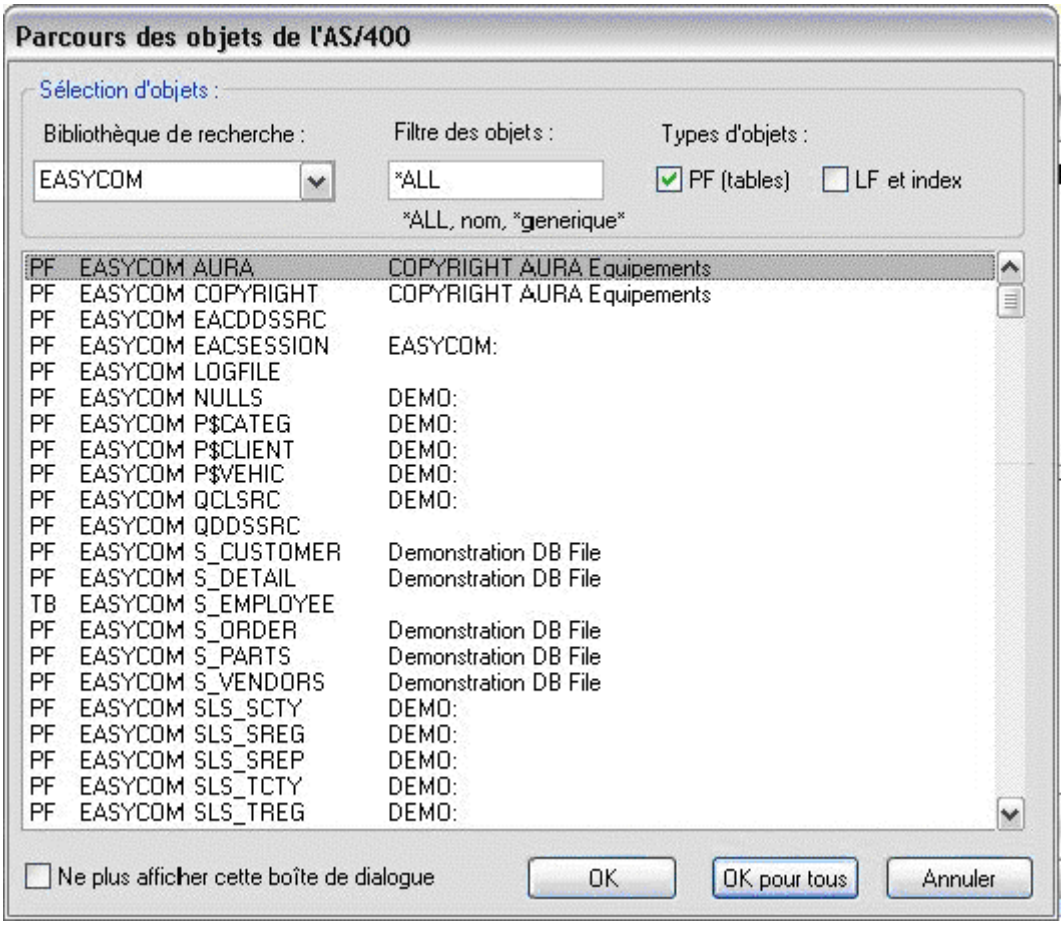
Note

La connexion par défaut utilisera certaines propriétés de ce profil, notamment sa LIBL.

La liste de bibliothèque peut être précisée dans les infos étendues de la connexion (option INITLIBL) ou bien côté AS/400.

Il est possible d'adopter les droits d'un autre profil une fois connecté par la fonction ASUtilisateur.

L'importation va créer une connexion dans l'analyse et les fichiers importés seront par défaut associés à cette connexion.



Bibliothèque de recherche

Cette zone permet de choisir le nom de la bibliothèque AS/400 qui contient les fichiers dont vous voulez importer la description.

*USRLIBL : Bibliothèques accessibles par le profil utilisateur.

Filtre des objets

En important un fichier physique :

- Chaque fichier logique lié au fichier physique sera associé à une clé,
- Une clé composée (index sur plusieurs rubriques) apparaît comme une rubrique supplémentaire dans la description du fichier.

En important un fichier logique :

- La description du fichier reprend les rubriques du physique mais la seule clé est celle du logique.

Il est possible d'importer des logiques avec des SELECT / OMIT.

Dans la liste les différents types de fichier sont proposés :

- PF - fichier physique,
- LF - fichier logique,
- TB - table (SQL),
- IX - index (SQL),

Noms des fichiers

Il est possible de contrôle la casse (majuscules / minuscules) des fichiers importés avec l'option LITERALCASE des Infos Etendues de la connexion, à définir avant l'importation.

Attention à ce que les noms de fichiers ou de rubriques importés ne soient pas des mots réservés de WinDev.

Attention

La liste de bibliothèque est la LIBL du profil, si le fichier se trouve dans cette liste les infos étendues du fichier ne reprennent que le nom du fichier (sans bibliothèque).

Si la bibliothèque n'est pas dans la LIBL du profil mais qu'elle est saisie dans la zone "Bibliothèque de recherche", le nom de la bibliothèque sera également repris dans les infos étendues du fichier.

Il est possible de compléter la liste de bibliothèques initiales d'une connexion par l'option INITLIB des infos étendues de la connexion.

Sélection des tables à importer

Cet écran reprend les fichiers précédemment sélectionnés, confirmer les tables dont vous souhaitez importer les descriptions.

Vous retrouvez également les descriptions de Programmes (*RPC) et de Data Queue (*DTAQ).

Contraintes

Les contraintes référentielles sont également importées sous la forme de liaisons de type 0,n à 1,1.

Les contraintes de clé primaire sont importées en tant que rubrique de type clé unique.

Importer des données

AS400 vers HyperFile

Pour importer l'intégralité des données d'un fichier AS400 dans un fichier Hyperfile, sélectionnez l'option "Convertir les données dans le format Hyper File Classic" dans l'assistant de création de fichier.

Hyperfile vers AS400

L'exportation des fichiers d'une analyse se fait par un outil spécifique, le constructeur de DDS, qui se trouve dans le menu Easycom For WinDev 2026 - Exportation de fichiers vers l'AS/400.

Dans ce constructeur de DDS, vous disposez d'une option "Transférer les données" pour dupliquer les enregistrements.

La duplication peut se faire en même temps que la création du fichier ou sur un fichier existant.

Il ne s'agit pas d'un outil de synchronisation mais d'un transfert de tous les enregistrements du fichier Hyper File vers le fichier AS400.

Accès natif

Les bases

L'accès natif permet de manipuler des fichiers AS/400 avec les fonctions classiques Hyperfile avec cependant certaines restrictions (voir spécificités AS/400) et quelques considérations particulières, notamment :

- la bibliothèque et le chemin d'accès,
- les verrouillages,
- les transactions,

Avant toute opération il faut ouvrir une connexion. Cette connexion peut être automatique si elle est définie dans l'analyse ou avec la gestion SSO ou explicite

WDMAP est entièrement compatible avec les fichiers AS/400, vous pouvez l'utiliser pour consulter, éditer et ajouter des enregistrements.

WDETATS et les éditions sur des fichiers AS/400 se manipulent exactement de la même manière que pour des fichiers Hyperfile ou d'une autre base. Il peut s'agir de fichiers, de requêtes, de vues, etc...

```
HExécuteRequête(Requete1,MaConnexion1,hRequêteSansCorrection, "'C-05'")
iAperçu(i100, "Aperçu écran", Vrai)
// Exécution de l'état
iImprimeEtat(Etat_Requête1)
```

HCréation n'est pas supporté, HCréationSiInexistant ne crée pas le fichier mais provoque son ouverture.

Un fichier AS/400 s'ouvre comme un fichier Hyper File par la fonction HOuvre. Il se parcourt selon une clé qui correspond à un fichier logique : HLitPremier(SP_CUST, CUST_ID)

De même un filtre sur une clé va définir un intervalle sur le fichier logique, un filtre avec condition sera géré par un OPNQRY.

Il ne faut pas utiliser le numéro d'enregistrement pour une lecture directe (HLit) sur le numéro d'ordre mais sur le numéro relatif (RRN). Mais HSauvePosition et HRestorePosition fonctionnent bien.

HChangeNom n'est pas supporté.

Pour sélectionner un fichier dans une bibliothèque particulière, utiliser AsExec avec les commandes de gestion de bibliothèque (CHGCURLIB, ADDLIBLE ou RMVLIBLE) . Voir le chapitre sur les bibliothèques.

Les opérations de lecture mettent à jour les variables telles que hEnDehors ou hTrouve.

Les erreurs éventuelles mettent à jour les variables hErreurIntégrité, hErreurBlocage ou hErreurDoublon.

HErreurInfo donne le détail d'une éventuelle erreur Hyperfile.

ErreurInfo donne le détail d'une éventuelle erreur sur une fonction Easycom.

Spécificités de l'AS/400

Certaines fonctionnalités de WinDev ne peuvent avoir de correspondance avec le fonctionnement d'un AS/400.

Longueur des noms de fichiers

Sur l'AS400 un nom de fichier est limité à 10 caractères mais Easycom utilise les alias SQL pour gérer les noms longs (jusqu'à 128 caractères pour les fichiers et 30 caractères pour les rubriques).

Lorsque vous exportez des fichiers de plus de 10 caractères avec des débuts de noms communs, un système de "chaîne de transformation" dans le constructeur de DDS permet de contrôler précisément les noms systèmes de tous les fichiers à exporter.

En importation de DDS, l'option LITERALCASE des infos étendues de la connexion vous permet de contrôler la casse (minuscules/majuscules) des noms de fichiers à importer.

Numéro d'enregistrement

La lecture directe d'un numéro d'enregistrement va utiliser le RRN (Relative Record Number) qui n'est pas le numéro d'ordre de l'enregistrement mais un numéro relatif.
La fonction HLit est donc fortement déconseillée.

Enregistrement rayé

Cette notion Hyperfile n'existe pas sur l'AS400.

Lecture bloquante

L'AS400 ne permet qu'un seul enregistrement verrouillé par utilisateur ou le blocage de l'intégralité du fichier.
La fonction `HBloqueFichier` ou un `HCréeVue` avec l'option *`hVueBlocage`* sont possibles.

Le temps de réponse sur une tentative d'accès bloquant d'un enregistrement verrouillé dépend de la propriété du fichier AS400 (WAITRCD) et de la valeur du h.NbEssais (voir verrouillages).

`HDEbloqueNumEnr,` avec l'accès natif, débloque l'enregistrement verrouillé qu'il soit courant ou non, et quel que soit le paramètre envoyé. Ce comportement vient du fait que l'accès natif AS/400 ne sait verrouiller qu'un enregistrement à la fois.

Transactions et journalisation

La gestion des transactions nécessite que tous les fichiers utilisés dans la transaction soient **journalisés** (voir gestion des transactions).

Blobs

Les champs mémos HyperFile peuvent être gérés comme des champs de type BLOB (Binary Large Object) en important des DDS.

Cette gestion est assez contraignante (les fichiers doivent être journalisés et il faut utiliser les transactions).

L'export de fichiers vers l'AS400 (mode DDS) avec des mémos utilise un système plus souple avec un fichier "mémo" qui reprend le nom du fichier physique suivi de "__" (deux underscores).

Fonctions non supportées

HCréation

HChangeNom

HEcrit

Les fonctions de gestion du journal (au sens WinDev). La gestion des transactions se fait par la fonction SQLTransaction, voir journaux et transaction.

HGèreJournal - HJournalInfo - HJournalRecrée - HJournalRedémarre - HJournalStop - HPoste...

HPositionne

HRaye et **HLibere** (Les champs rayés n'existent pas sur l'AS/400)

Fonctions limitées

HCréationSiInexistant : ne créé pas le fichier mais l'ouvre.

HDecritFichier : le fichier temporaire sera sur le PC.

HDecritRubrique : le fichier temporaire sera sur le PC.

HSecurite : la sécurité est toujours active sur l'AS/400.

HVersion : retourne toujours 0.

Autres remarques :

Une requête de requête n'est pas supportée.

Un filtre sur une requête n'est pas supporté.

Equivalences de types

Suivant le type de données AS/400 un type de données HF va correspondre automatiquement au moment de l'importation, de l'utilisation de SQL, ou encore lors de l'utilisation de HDeclareExterne.

Après importation un type HF compatible peut être utilisé par modification directe des infos étendues. De même qu'un type de données peut être choisi au moment de l'exportation. Bien entendu il est plus prudent de laisser les équivalences par défaut en cas de doute.

Voici la liste des types de données HF utilisés lors de l'importation :

Type DDS	Type SQL	Condition	Type HF
A (caractère)	CHAR		**Texte**
A OPTION(VARYING) (caractère variable)	VARCHAR		**Texte**
G + CCSID 13488 (Unicode)	GRAPHIC CCSID(13488)		**Texte Unicode (version 12 seulement)**
G + CCSID 13488 + OPTION(VARYING) (Unicode variable)	VARGRAPHIC CCSID(13488)		**Texte Unicode (version 12 seulement)**
P (décimal condensé) ou Z (décimal étendu)	DECIMAL ou NUMERIC	Entier, <=4 chiffres et pas de decimal	**Entier signé sur 2 octets**
		Entier, <=9 chiffres et pas de decimal	**Entier signé sur 4 octets**
		Entier, <=19 chiffres et pas de decimal	**Entier signé sur 8 octets**
		<= 38 chiffres et Nombre de decimal <=6	**Numérique**
		Autres cas	**Texte**
L	DATE		**Date**
T	TIME		**Heure (HHMMSS)**
Z	TIMESTAMP		**Date et Heure**
B4 (entier court)	SMALLINT		**Entier signé sur 2 octets**
B9 (entier long)	INT		**Entier signé sur 4 octets**
B19 (entier 64 bits)	BIGINT		**Entier signé sur 8 octets**
F	FLOAT		**Réel sur 4 octets**
F double précision	DOUBLE		**Réel sur 8 octets**
H	BINARY		**Chaîne binaire**
H	VARBINARY		**Chaîne binaire**
N/A	CLOB		**Mémo texte**
N/A	BLOB		**Autre mémo binaire**
N/A	DBCLOB + CCSID 13488		**Mémo Unicode**

Remarque : si le fichier importé a d'abord été exporté via l'outil « constructeur DDS », les types de données ne suivront pas nécessairement ce tableau afin de conserver le type d'origine (originellement dans l'analyse).

Le texte « HFTYPE=xx peut apparaître dans les fichiers exportés et correspond à la valeur numérique des constantes de type HF. (Par exemple pour un champ texte, on a : HFTYPE=2, ce qui correspond à la valeur de wlChaîne).

Le tableau ci-dessous est un résumé synthétique des associations de types.

Ces associations sont paramétrables à l'exportation de fichiers Hyper File vers l'AS/400 et automatiques en importation.

Le support d'unicode nécessite un CCSID approprié.

Le support des blobs nécessite des fichiers journalisés et une gestion des transactions.

Les dates et heures peuvent être associées à des formats alternatifs, dans ce cas l'option DATETYPE ou TIMETYPE des infos étendues de la connexion ou du fichier ou l'option NATIVETYPE des infos étendues de la rubrique sont nécessaires.

Type AS/400	Longueur AS/400	Décimale	Type Hyper File
CHAR / VARCHAR			Texte
UNICODE			Texte
SMALLINT	2 octets		Entier sur 2 octets
INTEGER	4 octets		Entier sur 4 octets
BIGINT	8 octets		Entier sur 8 octets
FLOAT			Réel sur 4 octets
DOUBLE			Réel sur 8 octets
DATE	10		Date sur 8
TIME	8		Heure
TIMESTAMP	26		DateHeure
BINAIRE (chaîne sans conversion)			Binaire
(fichier mémo)			Son, image, binaire sur 8
BLOB			Son, image, binaire sur 8
CLOB			Son, image, binaire sur 8
NUMERIC (étendu) / DECIMAL	<= 4	0	Entier sur 2 octets
	<= 9	0	Entier sur 4 octets
	<= 18	0	Entier sur 8 octets
	<= 15	De 1 à 15	Réel sur 8 octets
	<= 22	< = 5	Monétaire
	Texte		Texte

CCSID

Il existe en plus une page de code pour la traduction des données alphanumériques.

C'est le **CCSID** (Coded Character Set Identifier) qui est paramétré sur le système.

Vous retrouvez sa valeur par la commande :

```
DSPSYSVAL QCCSID
```

Un AS/400 français aura un CCSID égal à **1147** ou **297** mais il est possible que le système ne soit pas paramétré pour prendre en charge la conversion ; dans ce cas, le CCSID aura la valeur **65535**.

De plus, chaque fichier physique ou logique peut être compilé en utilisant un code autre que celui spécifié par le système.

Il est possible de connaître le CCSID de chaque fichier de données en utilisant la commande `DSPFFD BIB/FILE.`

En utilisant cette commande, c'est en regardant au niveau de **chaque champ** que l'on peut voir apparaître le jeu de caractères utilisé pour la compilation.

Conversion et formats particuliers

Champs et formats

Dates

Si la rubrique autorise les NULL, une date vide sera considérée comme telle, autrement une date à blanc sera codée avec les valeurs minimales en écriture (par exemple 01.01.0001).

En lecture, une date minimale AS/400 sera codée comme étant une donnée à blanc dans WinDev.

Remarque : un format date ou heure dans l'analyse peut être associé à d'autres types. L'option DATETYPE et TIMETYPE des infos étendues de la connexion ou du fichier ou l'option NATIVETYPE des infos étendues de la rubrique sont dans ce cas nécessaires pour certaines d'utilisations.

Le format pour les dates à utiliser dans les requêtes en mode *`HRequêteSansCorrection`* est le format ***ISO** : pour les dates : YYYY-MM-DD (2005-12-25) et pour les heures : HH:MM:SS (17:59:59).

Mémos et BLOB

La gestion des mémos est différente entre des fichiers AS/400 avec des blobs importés dans l'analyse et des fichiers Hyper File avec des mémos exportés vers l'AS/400.

Fichiers importés

Un champ de type BLOB sur le fichier physique sera associé à un champ de type mémo dans la description du ficher.

Attention, le fichier doit être journalisé et il faut utiliser la gestion de transaction.

Fichiers exportés

Un fichier spécifique est créé en reprenant les 8 premiers caractères du nom du fichier principal suivi de "__" (deux underscores). Dans le fichier principal le champ mémo contient l'identifiant du mémo dans le fichier annexe.

L'ouverture, l'écriture et la lecture sur le fichier mémo sont synchronisées avec l'ouverture, l'écriture et la lecture du fichier principal. S'il n'y a que des mémos textes, ils sont convertis en EBCDIC mais si le fichier comporte un autre type de mémos, ils sont tous stockés en hexadécimal (binaire).

La gestion est transparente en accès natif (`HLitPremier`, etc.…) mais dans les requêtes SQL en revanche on ne récupère pas le contenu mémo mais l'identifiant.

Si un traitement n'utilise pas les champs 'mémo' du fichier, vous avez intérêt à "débrancher" la gestion par la fonction `HGèreMémo` et le mode de gestion *`hMémoNon`*. Pour réactiver, utilisez le mode *`hMémoOui`*.

UNICODE

UNICODE est géré depuis la version 4.58.56 d'Easycom Serveur de manière automatique.

Il existe un cas de figure particulier cependant : dans le cas d'une requête SQL avec une valeur littérale au passage de paramètre vers un champ de type unicode sur l'AS/400, et donc une conversion PC vers AS, il faut que le JOB soit associé à un CCSID qui supporte l'unicode (1147, 13488…). Il est possible de modifier la configuration serveur d'Easycom (CFGEAC) pour spécifier un CCSID pour tous les jobs ou de modifier directement les propriétés du job en cours :

Pour reprendre le CCSID de l'utilisateur (en supposant qu'il supporte l'unicode) :

```
ASExec("CHGJOB CCSID(*USRPRF)")
```

Gestion des caractères spéciaux

Du coté du PC comme du coté AS/400, les caractères sont interprétés selon une page de code.

Sur le PC, on parle de **CodePage**, qui est le même pour l'Europe de l'Ouest et tout le monde occidental.

Sur l'AS/400 le fonctionnement est identique. Mais le stockage des données se fait selon le code **EBCDIC**.

Codepage

Par défaut les tables de caractères sont automatiquement gérées en fonctions des propriétés régionales et de la configuration de l'AS/400 mais il est possible de choisir explicitement un fichier de table de conversion différent. Il faut, pour remédier à cela, installer le "Pack international Easycom" disponible en téléchargement depuis le site.

1- Utilisez le fichier de table de conversion correct, exemple E037ansi.CPG

Pour appliquer la conversion à l'ensemble des connexions :

Modifiez le fichier easycom.ini

```
[GENERAL]
ConvTable=c:\Program Files\Easycom\WinDev10\E037ANSI.cpg
```

Pour appliquer la conversion à une connexion spécifique, ajouter l'option au niveau des infos étendues de la connexion :

```
<EASYCOM>
CODEPAGEFILE=c:\Program Files\Easycom\WinDev10\E037ANSI.cpg
</EASYCOM>
```

Recherches

Fichier logique

Lorsque vous utilisez les fonctions HLitRecherche(*Fichier*, *Champ*, *valeur*), c'est le fichier logique associé à la clé qui est ouvert et utilisé. Son nom est défini dans les infos étendues de la rubrique.

De même le résultat d'un HFiltre va déterminer la clé et donc le fichier logique utilisé pour parcourir le jeu d'enregistrements ou, dans le cas d'une condition, créer un OPNQRY.

Sur une requête

Attention, un HLitRecherche sur une rubrique de requête **non indexée** va devoir lire chaque enregistrement pour trouver la valeur recherchée. Ce traitement peut donc s'avérer très long d'une part et parfois même incohérent puisque les enregistrements ne sont pas nécessairement dans l'ordre.

L'option SQLIDX des infos étendues permet la création dynamique d'index sur toutes les rubriques d'une index ("loupes" dans les tables).

Clés composées

La recherche sur un clé composé se fait sur un fichier AS/400 de la même manière que sur un fichier Hyper File.

Il faut construire la valeur de la clé composée avec la fonction `hConstruitValClé` ou passer en paramètres les valeurs dans un tableau avec la syntaxe [valeur1, valeur2, valeur3...].

Si la clé est réduite, il faut compléter la borne minimale par `HValMin` et la borne maximale par `HValMax`.

Si une séquence de tri a été définie (sur le fichier ou dans les propriétés du job) et dans laquelle le dernier code n'est pas FF, utiliser l'option HFMAXKEY des infos étendues.

Clés composées réduites

Il est possible de supprimer des rubriques **à la fin** de la description de la clé composée, la recherche sur cette clé utilisera alors les champs restants. Mais il est plus simple de ne passer que les premiers paramètres en complétant les bornes par HValMin et HValMax.

Filtres

Contextes Hyper File indépendants

Un même fichier peut être utilisé dans plusieurs fenêtres avec l'option "Contexte Hyper File indépendant". Plusieurs chemins d'ouvertures sont alors utilisés. Le fichier peut avoir différents filtres, mais dans le cas particulier où les filtres ont une même clé avec des conditions différentes, l'option DUPPATH des infos étendues de la connexion doit être active.

Filtre simple

Un filtre simple (sur une clé avec une valeur minimale et une valeur maximale) va utiliser le fichier logique associé à cette clé. Pour une condition plus complexe c'est un OPNQRY qui sera fait.

Filtres sur une clé composée

Pour construire la valeur d'une clé composée, utilisez la fonction `HConstruitValClé`.

Si la borne minimale et la borne maximale sont identiques, et si tous les composants de la clé ne sont pas spécifiés, il est nécessaire de compléter les bornes par les constantes `hValMin` et `hValMax`.

L'exemple suivant permet de rechercher tous les enregistrements du fichier client correspondant à "Dupond" :

```
HFiltre(Clients,Nom,
HConstruitValClé(Clients, Nom, "Dupond")+hValMin, ...
HConstruitValClé(Clients, Nom, "Dupond")+hValMax)
```

Séquence de tri

De base, les séquences de tri avec l'AS/400 sont EBCDIC et diffère du tri ASCII :.

- l'ordre EBCDIC est : Minuscules, Majuscules, Chiffres.
- l'ordre ASCII est : Chiffres, Majuscules, Minuscules.

Pour retrouver une séquence naturelle sous WinDev, il faut créer ou recréer les chemins d'accès sur l'AS/400 avec la séquence de tri appropriée.

Exemple

```
CRTLF FILE(EASYCOM/SP_CUST_CU) SRTSEQ(*LANGIDSHR)
```

On peut aussi recompiler le membre du fichier source (QDDSSRC) en précisant la séquence de tri souhaitée.

De nombreuses tables sont disponibles sur l'AS/400, on peut les consulter par la commande :

```
QSYS/WRKTBL TBL(*ALL)
```

Par exemple **SRTSEQ(*LANGIDSHR)** crée une séquence de tri ou les chiffres sont en premier, les lettres minuscules, majuscules et accentuées sont mélangées, dans la langue configurée par le job.

Attention, si la rubrique utilise une séquence de tri dans laquelle le dernier caractère n'est pas le code "FF", voir l'option HFMAXKEY des infos étendues de la connexion.

Verrouillages

Attention, en dehors des transactions, de l'utilisation d'alias ou de contextes multiples, il n'est pas possible de bloquer plusieurs enregistrements sur un même fichier AS400 par un même profil mais il est désormais possible de bloquer l'intégralité d'un fichier ou d'une vue.

Dans le cas des contextes HyperFile, si l'option DUPPATH des infos étendues de la connexion est active, c'est un chemin d'accès dupliqué qui sera utilisé avec ses propres verrous.

Attente et nombre d'essais

Sur un AS/400, la gestion des blocages d'enregistrement diffère de celle de WinDev : il n'est possible de bloquer qu'un seul enregistrement par fichier ouvert et par utilisateur. Par contre, il existe une gestion de file d'attente de blocages pour chaque enregistrement.

Il existe une seconde différence : à chaque tentative de blocage infructueuse (tentative de verrouillage d'un enregistrement déjà verrouillé), l'AS/400 va attendre un certain temps la libération de l'enregistrement. Le compte rendu n'est donné qu'à la fin de ce délai.

Cette attente, par défaut 60 secondes, correspond au paramètre "WAITRCD" de chaque fichier, physique et logique, utilisé par WinDev. Pour voir la valeur de ce paramètre, utilisez la commande DSPFD.

Le temps de réponse en verrouillage d'un enregistrement déjà verrouillé dépend donc de deux paramètres :

- WAITRCD propriété du fichier AS400,
- H.NbEssais, propriété WinDev.

Vous pouvez modifier ce délai sur l'AS/400 au niveau du fichier physique, par la commande CHGPF :

```
CHGPF FILE(LIB/FICHIER) WAITRCD(10)
```
pour attendre 11 secondes

```
CHGPF FILE(LIB/FICHIER)
WAITRCD(*IMMED)
```
pas d'attente, réponse immédiate

Le constructeur de DDS propose par défaut de compiler les fichiers avec l'option WAITRCD(*IMMED).

`HModifie` doit verrouiller l'enregistrement

Pour faire une série de modifications, il est donc vivement recommandé d'ouvrir tout de suite le fichier en écriture.

Dans le cas contraire le `HModifie` va devoir relire l'enregistrement pour comparer les valeurs avec le cache. En cas de données modifiées (entre la première lecture et la vérification) :

Sous WinDev : une fenêtre affiche les données d'origine et les nouvelles en demandant la confirmation de la modification.

Sous WebDev où il n'est pas possible d'afficher ce type de fenêtre, la modification risque de provoquer une erreur. Il faut donc utiliser une gestion d'erreur (HSurErreur) ou l'option DRVOPTIMISTIC des infos étendues de la connexion.

Le code conseillé pour une série de modifications est donc :

```
TANTQUE PAS condition
 HLitSuivant(Fichier, hBlocageEcriture)
 // affecter valeurs...
 HModifie(Fichier)
FIN
```

Exemple :

```
    Countline est un entier
    HOuvreConnexion(MaConnexion)

    HLitPremier(Sp_cust,Cust_id)
    TANTQUE PAS HEnDehors(Sp_cust)
     Countline++
     Sp_cust.Firstname = Countline
     HSurErreur(Sp_cust,hErrTout,proc_erreur)
     HModifie(Sp_cust)
     HLitSuivant(Sp_cust,Cust_id)
    FIN


    //Procédure globale
    PROCEDURE proc_erreur()
    SELON HErreurEtatModification(Sp_cust,hEnrFichier)
     CAS hEtatActif
      //Forcer la modification
      RENVOYER opRéessayer

     CAS hEtatSup
      //On ajoute l'enregistrement et on annule la modification
      RENVOYER opAnnuler

     AUTRE CAS
      //On annule
      RENVOYER opAnnuler
    FIN
```

Règles d'intégrité (contraintes)

Les tests d'intégrité se font comme pour des fichiers Hyper File avec la fonction HErreurIntégrité après une mise à jour ou une insertion.

La fonction HGèreIntégrité ne permet pas de désactiver une contrainte AS/400.

Exportation des fichiers

Seules les liaisons de type 0,n -> 1,1 sont exportées par le constructeur de DDS.

Si vous avez des liaisons de type 1,0 à 1,1, le constructeur vous proposera de les transformer.

Les autres liaisons ne seront pas exportées.

Importation des fichiers

Les contraintes référentielles de l'AS/400 sont récupérées et importées sous la forme de liaison 0,n à 1,1.

Il est tout à fait possible d'ajouter des liaisons dans l'analyse à des fichiers AS/400 sans que la contrainte soit définie sur l'AS/400, c'est le moteur Hyperfile (côté PC) qui se charge alors de faire les lectures et contrôles avec bien sûr des performances moindres.

Journaux et transactions

Sur l'AS/400, la gestion de transactions est gérée au travers d'un journal d'activités placé sur l'AS/400.

Pour gérer les transactions sur un fichier AS/400, il faut auparavant créer un fichier destiné à recevoir le journal (CRTJRNRCV puis CRTJRN), puis démarrer la journalisation sur chaque fichier (STRJRNPF).

En mode natif il ne faut pas utiliser la fonction HTransactionDébut mais SQLTransaction.

La connexion doit bien sûr être ouverte avant le début de la transaction.

Les fonctions de WinDev `SQLTransaction` agissent directement dans le journal de l'AS/400.

Tous les fichiers concernés doivent être journalisés et la description de chaque fichier doit contenir dans les **infos étendues** l'option suivante :

```
<EASYCOM>
JOURNALED=TRUE
</EASYCOM>
```

Remarque : par défaut l'importation des DDS met automatiquement à jour cette option, il est possible de désactiver ce fonctionnement avec l'option AUTOJOURNAL des infos étendues de la connexion.

La transaction est annulée en cas d'erreur fatale ou d'arrêt du programme.

Commencer une transaction : `SQLTransaction(SQLDébut)`

Valider la transaction (COMMIT) : `SQLTransaction(sqlFin)`

Annuler la transaction (ROLLBACK) : `SQLTransaction(sqlAnnule)`

Isolation Level

Sur l'AS/400 plusieurs modes transactionnels sont possibles, ils déterminent notamment le niveau d'isolation (Isolation Level), c'est à dire si les changements effectués sont visibles ou pas par d'autres utilisateurs ainsi que le verrouillage (partage en écriture ou accès exclusif).

Par défaut les transactions WinDev sont toutes sur le même niveau, il correspond à la commande `STRCMTCTL LCKLVL(*CHG)`.

Chaque enregistrement devant être mis à jour (pour un fichier ouvert sous contrôle de validation) est verrouillé. Tout enregistrement modifié, ajouté ou supprimé reste verrouillé jusqu'à la prochaine validation ou invalidation de la transaction. Les enregistrements consultés pour mise à jour, mais non modifiés, sont déverrouillés.

Performances - conseils

Les deux éléments clés qui déterminent la rapidité de l'accès aux données sont :

- le nombre d'accès réseau,
- le volume des données transférées,

Configuration et easycom.ini

Utiliser l'écran "Optimisation" de Easycom Configuration.

En déploiement si vous n'avez pas fait l'installation complète, ces informations se retrouvent dans le fichier easycom.ini qui se trouve dans le répertoire de Windows et contient les principaux paramètres de fonctionnement.

On peut améliorer les performances en jouant sur les entrées suivantes :

```
[BUFFERS]
records=xx
size=yyyy
```

La ligne records=xx détermine la taille du cache côté PC. Une lecture par clé pourra ainsi charger le nombre d'enregistrements spécifié dans un même bloc en réduisant ainsi le nombre d'accès distants. La taille maximale est fixée par l'entrée size=yyyy avec un maximum de 64Ko (65535).

Attention ! Un cache trop important peut aussi ralentir l'application en effectuant des lectures non nécessaires.

Il est également possible de définir un cache au niveau du fichier par la fonction ASProprietes.

```
[TCP]
Compression=1
```

La compression TCP/IP est activée en mettant la valeur à 1 (désactivée par la valeur 0). Les données sont alors compressées avant leur envoi et le volume transféré est ainsi réduit.

SQLIDX

L'option SQLIDX qui permet d'avoir des loupes dans les tables liées à des requêtes peut être désactivée dans les infos étendues de la connexion si cette fonctionnalité n'est pas nécessaire. En effet cette option sollicite davantage le serveur par la soumission de requêtes SQL supplémentaires.

ASPropriete

La fonction ASPropriete permet de fixer des propriétés de cache (en lecture et en écriture).

Ouvertures

Afin que la première ouverture soit la plus pertinente il peut être utile de modifier les infos étendues du fichier et la ligne MAIN en indiquant le fichier logique le plus utilisé. Ceci économisera le temps d'ouverture du fichier physique.

Liste et combos

Si vous avez des listes et combos associées à des fichiers qui ne sont que rarement mis à jour (liste de types, de catégories...), utilisez des tableaux ou des copies locales plutôt que des liaisons sur le fichier d'origine.

Tables Fichier avec liaisons

Si une fenêtre affiche un fichier avec liaisons, le chargement de chaque ligne de la table va entraîner des lectures de toutes les rubriques de l'enregistrement correspondant sur chacun des fichiers liés. En plus de la multiplication des accès, on perd aussi le bénéfice de la lecture par bloc, puisque pour chaque donnée liée, WinDev effectue une lecture par clé.

Deux autres approches peuvent être conseillées pour ce type de table :

· Utiliser une requête qui utilise un SELECT avec des INNER JOIN.

· Créer sur l'AS/400 un logique joint et l'importer dans l'analyse.

Filtres

Les filtres avec condition génèrent un OPNQRYF, alors que les bornes génèrent des accès par clé, puis séquentiels.

Un filtre sur clé sera donc plus efficace et la création d'un nouveau fichier logique peut être pertinente.

Verrouillages

Verrouillez dès la lecture les enregistrements destinés à être modifiés.

Ceci évitera le mécanisme de contrôle qui va relire l'enregistrement pour vérifier qu'il n'y a pas eu de modifications depuis la première lecture.

HCréeVue

Un HCréeVue sur condition va provoquer une lecture séquentielle de tous les enregistrements qui seront ensuite testés côté client, ce qui revient donc à lire et à charger l'intégralité du fichier… Pour éviter ce mécanisme, la solution consiste à faire précéder le HCréeVue par un HFiltre sur une condition identique. Dans ce cas un OPNQRYF est effectué sur l'AS/400 et seules les données pertinentes sont transférées.

Un HCréeVue sur bornes en revanche ne gagnera pas à être précédé du HFiltre.

Mémos

Les mémos sont stockés dans des fichiers complémentaires, par défaut l'ouverture, l'écriture et la lecture sur le fichier principal sont synchronisées avec l'ouverture, l'écriture et la lecture du (des) fichier(s) mémo. Si un traitement n'utilise pas le fichier mémo, vous avez tout intérêt à « débrancher » la gestion par la fonction HGèreMémo.

```
<Résultat> = HGèreMémo([<Nom du fichier>, [<Nom de la rubrique>,]]
<Mode de gestion>)
```

Avec *hMémoOui* pour activer et *hMémoNon* pour désactiver.

Attention : ces fonctions doivent s'accompagner de la fermeture et de la réouverture du fichier pour être prises en compte.

Blob

Si vous manipulez des fichiers avec des blobs, il peut être utile d'éviter le chargement et le transfert de données volumineuses dans les traitements où ils ne sont pas requis.

Un logique sur le fichier sans le champ blob peut être créé et importé comme un fichier alternatif, on peut aussi utiliser une requête.

HDéclareExterne

La fonction HDeclareExterne permet d'utiliser par programmation un fichier d'une base externe **sans importer la structure du fichier dans l'analyse**. Le fichier est récupéré avec toute sa structure et ses clés.

Syntaxe avec connexion

Si une connexion a été déclarée et est ouverte, il suffit de passer le nom du fichier

```
bresult = HDéclareExterne( Fichier, AliasFichier, Connexion)
```

Le nom du fichier est le nom du chemin d'accès (BIBLIOTHEQUE/FICHIER ou FICHIER) et le nom alias le nom qui pourra être utilisé dans la suite du code.

La connexion doit bien sûr être définie, soit dans l'analyse, soit par programmation et être de type Accès Natif AS400.

```
HDéclareExterne("EASYCOM/SP_CUST", Customer, MaConnexionAS)
```

Syntaxe sans connexion décrite

La fonction HDéclareExterne peut également contenir les paramètres de la connexion :

```
bResultat = HDéclareExterne( Fichier, Alias, Utilisateur, Passe,
AdresseIP, hAccèsNatifAS400 [,Accès])
```

Par exemple :

```
HDéclareExterne("EASYCOM/SP_CUST", "Customer", "QPGMR", "PASSWD",
"194.206.105.111", hAccèsNatifAS400)
```

Exemple

```
//Déclaration des variables
MaConnexion est une Connexion
sp_cust est une Source de Données
//Description de la connexion
SI
PAS      HDécritConnexion(MaConnexion,"user","pwd","194.206.160.105"
,"",hAccèsNatifAS400,hOLectureEcriture,"")
ALORS
      Info(HErreurInfo())
FIN

//Déclaration du fichier externe SP_CUST
SI PAS HDéclareExterne("EASYCOM/SP_CUST", SP_CUST, MaConnexion)
ALORS
      Info(HErreurInfo())
FIN
//Lecture des enregistrements du fichier SP_CUST
HLitPremier(SP_CUST)
TANTQUE PAS HEnDehors
 Info(sp_cust.cust_id,sp_cust.firstname,sp_cust.lastname,sp_cust.add
ress,sp_cust.city,sp_cust.state)
 HLitSuivant(sp_cust)
FIN
```

Utilisation conjointe de HF et AS/400

Faire un projet Dynamiquement HF et/ou AS/400

Le procédé repose sur l'utilisation des fonctions HOuvreConnexion, ou HDéclareConnexion, ainsi que HChangeConnexion et éventuellement HDeclareExterne.

Dans un premier temps, importer dans l'analyse les fichiers qui viennent de l'AS/400.

Le point le plus important est de conserver les fichiers AS/400 en type « AS400 » dans l'analyse, même s'ils sont plus souvent utilisés en mode HF. C'est pour raison pratique : les infos étendues ne sont visibles que si le fichier est en type AS/400.

Pour cela, il faut simplement les garder attachés à une connexion AS/400 au niveau de l'analyse.

Au moment de l'exécution, utilisez un appel à HDecritConnexion pour décrire la connexion en HF ou en AS/400, puis utilisez HChangeConnexion pour les fichiers concernés.

Le « test » de requête pourra également fonctionner en fonction des critères, celui-ci exécutant le code de projet (sous réserve que les HChangeConnexion soient implémentés lors du code projet). La fonction EnModeTest() peut être utile pour déterminer si l'on est en mode test au niveau du code du projet (profil de test, …)

Remarques : Dans ce cadre, la connexion déclarée dans l'analyse n'est jamais utilisée, et ne sert que pour le RAD, l'éditeur de requêtes, etc. Cette méthode n'est applicable que si le(s) développeur(s) ont accès à l'AS/400 en permanence ou n'utilisent pas le « livedata ».

Bibliothèques et objets AS/400

Bibliothèques

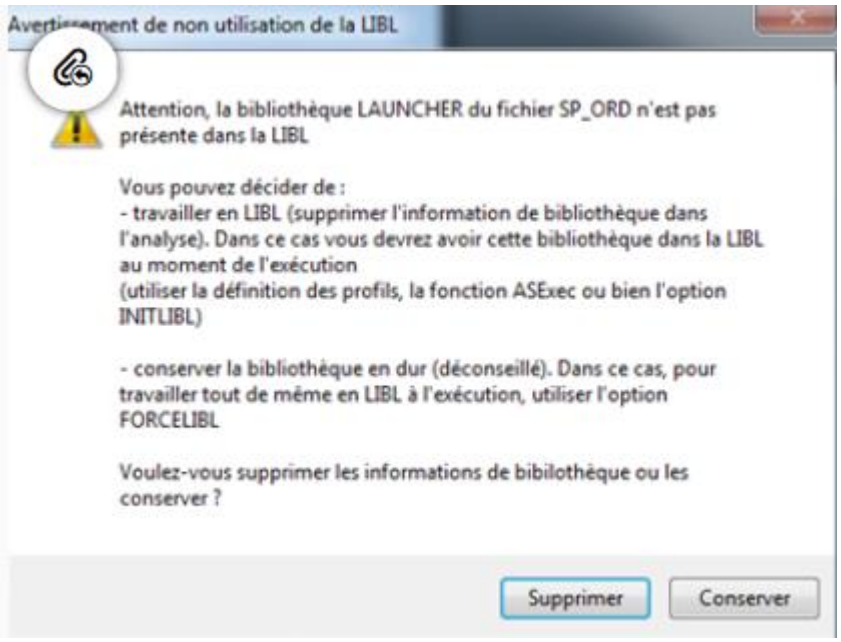
Ce point est particulièrement important pour comprendre et contrôler où se trouvent les fichiers ouverts sur l'AS400.

Import fichiers dans l'analyse Windev

Au moment de l'importation des fichiers dans l'analyse, la liste des fichiers par défaut correspond à la LIBL de l'utilisateur. Les fichiers sélectionnés directement et synchronisés ensuite pour mettre à jour les infos étendues ne comportent pas de nom de bibliothèque, ils seront recherchés dans la LIBL (voir ci-dessous).

Mais si vous tapez le nom d'une bibliothèque dans l'utilitaire d'importation de fichiers, le nom de cette bibliothèque sera ajouté aux infos étendues du fichier (option MAIN), si vous choisissez « Conserver ». Voir Bibliothèque explicite.

Ce sera donc toujours le fichier de cette bibliothèque qui sera ouvert.



Si vous choisissez « Supprimer », il n'y aura pas d'information sur la bibliothèque dans les infos étendues. Voir Bibliothèque implicite.

Bibliothèque explicite

Le nom de la bibliothèque peut être spécifié explicitement :

- dans les infos étendues du fichier (MAIN) ou de la clé (LF),
- dans les requêtes SQL (en mode *HRequêteSansCorrection*).

Bibliothèque implicite

Si le nom de la bibliothèque n'est pas explicitement renseigné dans les infos étendues, il va dépendre :

- de la liste des bibliothèques initiales éventuellement renseignées dans easycom.ini ou
- de la liste des bibliothèques initiales des infos étendues de la connexion (INITLIBL),
- des librairies issues de la JOBD de l'utilisateur,
- des librairies issues de EACJOBD (si différente de *NONE),
- de la librairie EASYCOM.

(voir aussi LIBL par défaut)

Dans ce cas il peut être géré en programmation par l'envoi de commandes :

```
AsExec("ADDLIBLE nom_biblio"),
AsExec("RMVLIBLE nom_biblio")
```

ou toute autre commande de gestion des bibliothèques (CHGCURLIB).

Le changement ne s'applique qu'aux fichiers ouverts après l'opération.

OVRDBF

Il est également possible d'utiliser la commande OVRDBF sur un fichier en particulier, sans oublier qu'une lecture ou un filtre se feront toujours sur le logique associé.

Ainsi un OVRDBF sur le physique uniquement ne s'appliquera pas à une lecture séquentielle qui utilise un fichier logique et qui doit donc faire également l'objet d'un OVRDBF.

```
ASExec("OVRDBF FILE(BIBLIO1/FICHIER1) TOFILE(BIBLIO2/FICHIER2)
MBR(MEMBRE) OVRSCOPE(*JOB)")
```

Bibliothèque et requête SQL en mode HRequêteDéfaut

Seul le nom du fichier est passé dans la requête, il sera recherché dans le LIBL du job. Pour choisir la bibliothèque le mieux est de faire précéder l'exécution par un CHGCURLIB via la fonction ASExec.

```
ASExec("CHGCURLIB MABIBLIO")
HExécuteRequête(REQ_Requête3,hRequêteDéfaut)
```

Fichiers AS/400

Fichiers physiques et logiques

Les données d'un AS/400 sont stockées dans des fichiers dits "physiques", ou "PF", ceux-ci ne possèdent qu'un seul index (ou clé) au maximum.

Il existe un second type de fichier de données : les fichiers "logiques", ou "LF". Ils ne contiennent aucun enregistrement mais "pointent" sur un fichier physique. Ces fichiers peuvent être considérés comme des vues, complètes ou partielles, sur un fichier physique.

WinDev "voit" les fichiers physiques comme des fichiers de données.

En revanche les fichiers logiques peuvent être "vus" comme une clé (simple ou composée) d'un fichier Hyper file ou bien comme un fichier Hyper File à part entière.

On va donc retrouver le nom d'un fichier physique dans les infos étendus du fichier et le nom du ou des logiques dans les infos étendues des rubriques mais il est possible d'importer directement un fichier logique.

Par ailleurs, dans l'éditeur, une clé composée apparaît comme une rubrique supplémentaire.

En fonction de ces types d'accès il existe certaines contraintes :

Fichiers logiques vus comme une clé

- La description des champs du logique doit correspondre exactement à celle du physique associé.
- Les repositionnements, les concaténations et les limitations de champs sont proscrites.
- Les sélections et omissions sont possibles mais le développeur et l'utilisateur doivent en être avertis, afin qu'ils ne cherchent pas à utiliser un enregistrement exclu de la sélection ou de l'omission.
- Les renommages des champs sont possibles. Si un champ clé est renommé, il faut alors faire la modification de façon manuelle dans WinDev et sur l'AS/400.
- Les logiques multi-formats, les fusions et les jointures sont proscrits.

Fichiers logiques vus comme un fichier Hyper File

• Les sélections et omissions sont possibles mais le développeur et l'utilisateur doivent en être avertis, afin qu'ils ne cherchent pas à utiliser un enregistrement exclu de la sélection ou de l'omission.

• Les logiques multi-formats et les fusions sont proscrits.

• Seules les jointures mono-format sont possibles.

Fichiers joints

La constitution de fichiers logiques joints sur l'AS/400 est une bonne alternative aux problèmes de performances sur des réseaux à bas débit.

Les avantages du fichier logique joint sont :

- L'AS/400 établit les liaisons entre les fichiers au moment de la lecture, et limite ainsi le nombre d'échanges avec le PC.
- On profite des fonctions de lecture anticipée d'EASYCOM.
- Les chemins d'accès existants pour les autres logiques sont réutilisés.
- La jointure est dynamique, faite au moment de la lecture, et n'entraîne pas de charge à l'écriture.

La création d'un fichier logique joint se fait en créant une source DDS compilée par un CRTLF.

Exemple :

Pour un fichier logique joint, la clé de recherche appartient au 1er fichier seulement.

```
      **************  Début des données  ********************************************************
0001.00                R RFMT                      JFILE(ANDRE2/SP_CUST ANDRE2/SP_ORD +     010613
0001.01                                            ANDRE2/SP_DET)                           010613
0002.00                J                           JOIN(1 2)                                010613
0003.00                                            JFLD(CUST_ID CUST_ID)                    010613
0003.01                J                           JOIN(2 3)                                010613
0003.02                                            JFLD(ORDER_ID ORDER_ID)                  010613
0003.03                  CUST_ID                   JREF(1)                                  010613
0004.00                  FIRSTNAME                                                          010613
0005.00                  LASTNAME                                                           010613
0006.00                  ORDER_ID                  JREF(2)                                  010613
0007.00                  SHIP_DATE                                                          010613
0008.00                  PART_NUM                                                           010613
0008.01                  UNIT_WT                                                            010613
0008.02                  UNIT_PRICE                                                         010613
0008.03                  QUANTITY                                                           010613
0009.00                K CUST_ID                                                            010613
0009.01                                                                                     010613
0010.00                                                                                     010613
      ***************  Fin des données  *********************************************************
```

Attention :

Un fichier logique joint n'est accessible qu'en lecture seule.

Le numéro d'enregistrement retourné après une lecture d'un fichier logique joint n'a pas de rapport avec celui du fichier physique sur lequel il est construit. Il faut donc utiliser la clé de recherche pour accéder au fichier de base à partir de la jointure.

Comment créer un Fichier Joint ?

Le fichier joint permet d'augmenter les performances d'accès aux différents fichiers, il est tout particulièrement adapté aux tables avec des fichiers reliés et apporte des gains de performances très intéressants.

Sur l'AS/400, taper STRPDM, option 3 - Gestion des membres.

Fichier : QDDSSRC / Bibliothèque : nom_biblio

F6 pour créer un nouveau fichier

Membre source : *nom_du_fichier*

Texte 'descriptif' : …

Entrer la description ; voir ci-dessous, jointure sur 3 fichiers

F3 pour quitter, enregistrer O pour Oui

F14 pour compiler le membre

Attention !!! Un fichier joint n'est accessible qu'en lecture seule.

Fichiers logiques OMIT et SELECT

Les logiques avec omission ou sélection ne sont pas considérés comme des index de fichiers physiques. Ils ne sont donc pas automatiquement importés avec le fichier physique mais on peut rajouter manuellement ces fichiers comme index dans l'analyse, dans les 'infos étendues' par :

```
<EASYCOM>
LF=nom_du_logique_sur_AS/400
</EASYCOM>
```

Mais on peut aussi les considérer comme des fichiers 'maître' n'ayant qu'un seul index.

Il suffit d'en importer leur structure dans l'analyse en cochant "LF" dans la boîte de dialogue.

Attention à l'utilisation de ces fichiers : Il peut y avoir une perte d'enregistrements du à la définition de sélection – omission.

Remarque : Si la structure des fichiers n'a pas été importée dans l'analyse, faites un HdeclareExterne sur le logique.

Dans ce cas, il n'y a pas de lien entre le fichier physique et le fichier logique, il faut donc faire des HSauvePosition et HRestaurePosition.

Fichiers DDM

DDMF est le système réseau de l'AS/400. Easycom gère les fichiers DDMF.
On ouvre sur un AS/400 un fichier qui est physiquement sur un autre, avec évidemment des temps d'accès moins bons.

Ces fichiers sont gérés comme tout autre fichier.

Lors de l'importation des fichiers AS/400 dans une analyse WinDev, il n'est pas possible de sélectionner l'attributs DDMF.

Pour sélectionner un fichier DDM, il faut utiliser la fonction HDeclareExterne.

Exemple :

```
Mon_fic_ddm est une Source de Données
SI PAS HDéclareExterne("LIB/MONDDM", mon_fic_ddm, Maconnexion) ALORS
 Info(HErreurInfo())
 Retourner Faux
FIN
HLitPremier(mon_fic_ddm, clé)
```

Fichiers multi-formats

Easycom gère les fichiers multi-formats. Par défaut c'est toujours le premier format qui est utilisé.

Pour accéder à un format particulier, il faut préciser le nom du format dans l'option MAIN des infos étendues du fichier ou l'option LF des infos étendues de la rubrique clé selon la syntaxe suivante :

```
BIBLIO/FICHIER *RFORMAT=NOM_DU_FORMAT
```

ou

```
FICHIER NOM *RFORMAT=NOM_DU_FORMAT
```

Exemple d'infos étendues à renseigner sur le fichier physique et les logiques :

```
<EASYCOM>
```

```
MAIN=BIBLIO/FICHIER *RFORMAT=NOM_DU_FORMAT
</EASYCOM
```

et

```
<EASYCOM>
LF=BIBLIO/FICHIER *RFORMAT=NOM_DU_FORMAT
</EASYCOM
```

Remarque : Pas de possibilité de requête

Fichiers 36

Easycom permet de travailler sur les fichiers 36 de l'ancien système.

Si le fichier 36 a un IDDU (Interactive Data Definition Utility) associé, il est vu comme un fichier AS/400.

S'il n'y a pas de description :

- Il faut créer un fichier AS/400 (vide) qui a la description désirée.
- Le nom du fichier dans la programmation est de la forme `*FMT=nom_du_fichier_de_format`

Exemple

Nous avons dans la bibliothèque "LIB36" le fichier "FIC36"

S'il n'y a pas de description :

- Il faut créer un fichier AS/400 (vide) qui a la descrition désirée.

Exemple : Créons sur l'AS/400 dans la bibliothèque "DEV" le fichier "FICVIDE"

Importer ce fichier "FICVIDE" dans l'analyse WinDev au format AS/400.

- Dans les "Infos étendues" du fichier et éventuellement sur chacune des clés, modifiez la ligne : MAIN =, en ajoutant *FMT = nom_du_fichier_de_format à la suite de fichier(36)

Exemple des 'infos étendues' du fichier :

```
<EASYCOM>
MAIN=LIB36/FIC36 *FMT=DEV/FICVIDE
</EASYCOM>
```

Exemple des 'infos étendues' des clés :

```
<EASYCOM>
LF=LIB36/FIC36 *FMT=DEV/FICVIDE
</EASYCOM>
```

Remarque : Pas de possibilité d'exécuter une requête SQL

Comment récupérer les fichiers 36 multi-formats ?

Les fichiers 36 multi-formats doivent être traités comme des fichiers 36. Il faut effectuer la même opération décrite ci-dessus pour chaque format (c'est-à-dire, il faut importer dans l'analyse, tous les formats).

WebDev : spécificités accès natif (verrouillages)

WebDev : spécificités accès natif (verrouillages)

Si vous avez créer un projet WebDev en Hyper File et que vous souhaitez le transformer en projet AS/400, deux points importants sont nécessaires.

En WinDev, l'implémentation de cette procédure n'est pas obligatoire.

1. Il faut créer une connexion AS/400.

En début de projet, ouvrir la connexion AS/400 par la fonction : HOuvreConnexion :

```
HOuvreConnexion(MaConnexion)
```

2. Implémenter une procédure pour la modification d'enregistrement

Exemple :

```
Countline est un entier
HOuvreConnexion(MaConnexion)

HLitPremier(Sp_cust,Cust_id)
TANTQUE PAS HEnDehors(Sp_cust)
 Countline++
 Sp_cust.Firstname = Countline
 HSurErreur(Sp_cust,hErrTout,proc_erreur)
 HModifie(Sp_cust)
 HLitSuivant(Sp_cust,Cust_id)
FIN


//Procédure globale
PROCEDURE proc_erreur()
SELON HErreurEtatModification(Sp_cust,hEnrFichier)
 CAS hEtatActif
  //Forcer la modification
  RENVOYER opRéessayer

 CAS hEtatSup
  //On ajoute l'enregistrement et on annule la modification
  RENVOYER opAnnuler

 AUTRE CAS
  //On annule
  RENVOYER opAnnuler
FIN
```

SQL

Création de requêtes

Une requête peut être créé soit par l'assistant soit par saisie directe (ou copie...) du code SQL. Suivant le code de la requête, le mode `HRequêteSansCorrection` devra être utilisé (notamment si des noms de bibliothèque ou des ordres spécifiques sont présents), il faut dans ce cas spécifier la connexion dans la syntaxe de la requête.

Les requêtes de sélection "SELECT" vont retourner une source de donnée qui peut être parcourue et manipulée en programmation.

Remarque : il est possible de remplacer l'option MAIN des infos étendues du fichier ou l'option LF d'une clé par une requête SELECT dont les colonnes correspondent à la description du fichier.

`hModifieFichier` peut être utilisé si l'option SQLLIVE est active (valeur 1 ou 3) dans les infos étendues de la connexion.

Dans ce cas les données du résultat de la requête sont modifiables même si elles combinent plusieurs fichiers.

Par défaut des index sont créés sur chaque colonne de la requête, ce qui permet notamment la présence des loupes dans les tables. Cette fonctionnalité peut être désactivée par l'option SQLIDX des infos étendues de la connexion.

Les contrôles d'intégrité de l'AS/400 sont évidemment respectés.

Il n'est pas possible d'utiliser les indices de tableau dans une requête SQL.

Pour définir une condition sur une clé composée dans une requête SQL, il faut préciser les conditions de chacune des composantes de la clé.

Les filtres sur les requêtes ne sont pas supportés.

La fonction HExécuteRequete envoie et exécute directement une requête créée par l'éditeur, cette requête peut contenir des paramètres ou pas, elle peut être de type SELECT ou UPDATE, INSERT, DELETE...

La fonction HExécuteRequêteSQL permet d'envoyer directement des requêtes sous la forme d'une chaîne SQL et d'être utilisé conjointement avec la fonction HPrepareRequêteSQL, voir requêtes préparées.

Voir exemple Requêtes et SQL.

Requête et transactions

Les requêtes AS/400 peuvent s'inscrirent dans les transactions. Une transaction peut tout à fait concerner des modifications par les fonctions classiques (HModifie...) et par des ordres SQL.

```
schaine est une chaîne
marequete est une Source de Données

SQLTransaction(sqlDébut,MaConnexion)
schaine="UPDATE SP_CUST SET Firstname='Jean' WHERE CUST_ID='C-01'"
HExécuteRequêteSQL(marequete,MaConnexion,hRequêteDéfaut,schaine)
SQLTransaction(sqlAnnule,MaConnexion)
```

Utilisation du mode HRequêteDefaut

Le mode HRequêteDefaut est utilisé pour un code de requête simple lorsque les fichiers concernés se trouvent dans l'analyse.

Il faut être très vigilant à la notion de bibliothèque puisque seul le nom du fichier apparaît dans la chaîne SQL. Il sera donc recherché dans la LIBL du job.

Le plus simple est de faire précéder l'exécution d'un CHGCURLIB :

```
ASExec("CHGCURLIB MABIBLIO")
HExécuteRequête(REQ_Requête3,hRequêteDéfaut)
```

Utilisation du mode HRequêteSansCorrection

Le mode HRequêteSansCorrection permet de ne pas interpréter la requête par WinDev avant envoi sur l'AS/400. Il permet également de travailler sur des fichiers non importés dans l'analyse en utilisant le chemin complet (BIBLIO/FICHIER).

Ce mode est utilisé dans le cas où le code SQL de la requête est modifié manuellement (exemple : ajout du nom de la bibliothèque) ou pour un code de requête compliqué (concaténation …).

Dans ce mode aucune information des infos étendues n'est récupérée (notamment le chemin d'accès).

HRequêteDéfaut **HRequêteSansCorrection**

Détermination automatique de la connexion associée aux fichiers présents dans la requête.	La connexion à utiliser doit être précisée dans la fonction HExécuteRequêteSQL.
Remplacement de tous les signes propriétaires PC SOFT (exemple : ']=' commence par) par leur équivalent en SQL standard.	Aucun remplacement n'est effectué. Il est nécessaire d'utiliser les signes SQL standard.
Formatage des dates et des heures selon le format utilisée par la base de données utilisée.	Aucun formatage n'est effectué. Il est nécessaire d'utiliser le format reconnu par la base de données.
Formatage des flottants (le séparateur de décimal peut être '.' ou ',')	Aucun formatage des flottants n'est réalisé.
Selon la base de données utilisée, remplacement des noms d'alias par les noms complets des rubriques dans les clauses Where, Order by et Group by	Aucun remplacement n'est effectué. Il est nécessaire d'utiliser directement dans le code de la requête les noms complets des rubriques dans les clauses Where, Order by et Group by.

Heures et dates en paramètres

En mode HRequêteDéfaut les dates et heures sont automatiquement formatées.

```
dToday est une Date
HExécuteRequête(REQ_Requête3,hRequêteDéfaut,dToday)
```

Si la valeur de date est passée directement il faut la mettre sur 8 caractères :

```
HExécuteRequête(REQ_Requête3,hRequêteDéfaut,"20050101")
```

Dans le cas particulier où des rubriques Date ou Heure sont associées à des types différent, l'option DATETYPE/TIMETYPE des infos étendues de la connexion ou du fichier et l'option NATIVETYPE des infos étendues de la rubrique permettent un formatage spécifique par Easycom.

Si le mode HRequêteSansCorrection est utilisé, les dates et les heures doivent être au format ***ISO** :

pour les heures HH:MM:SS,

pour les dates YYYY-MM-DD,

Exemple

```
HExécuteRequête(Nom_requête, Nom_connexion, hRequêteSansCorrection,
"'00:00:03'")
HExécuteRequête(Nom_requête, Nom_connexion, hRequêteSansCorrection,
"'1970-07-01'")
```

Syntaxe {Param} et formatage des chaînes

La requête contient en dur le nom de chaque paramètre. Ils doivent alors être initialisés avant l'appel ou bien passés dans l'ordre dans la fonction.

Attention, dans le cas des chaînes (texte), les quotes ne sont pas automatiquement insérées et doivent être ajoutées avant et après les valeurs :

```
REQ_Exemple2.Param1="'"+SAI_P1+"'"
```

Paramètres de la requête non précisés

Quelle que soit la façon de passer les paramètres à la requête, tous les paramètres de la requête ne doivent pas obligatoirement être précisés. Les conditions de la requête utilisant des paramètres non précisés seront ignorées.

Exemple : Soit la requête "Clients_nom_prénom" dont le code SQL est le suivant :

```
SELECT * FROM CLIENT WHERE NOM = {Param1} ET PRENOM = {Param2}
```

Si les 2 paramètres sont donnés :

```
HExécuteRequête(Clients_nom_prénom, hRequêteDéfaut, "Dupond", "Jean" )
```

exécutera la requête

```
SELECT * FROM CLIENT WHERE NOM = 'Dupond' ET PRENOM = 'Jean'
```

Si seul le nom est donné

```
HExécuteRequête(Clients_nom_prénom, hRequêteDéfaut, "Dupond")
```

exécutera la requête

```
SELECT * FROM CLIENT WHERE NOM = 'Dupond'
```

Si seul le prénom est donné. Le nom doit cependant être spécifié et correspondre à NULL.

```
HExécuteRequête(Clients_nom_prénom, hRequêteDéfaut, NULL, "Jean" )
```

exécutera la requête

```
SELECT * FROM CLIENT WHERE PRENOM = 'Jean'
```

Requêtes préparées

HPrépareRequêteSQL

Cette fonction Initialise une requête écrite en langage SQL et déclare cette requête au serveur de base de données pour optimiser les prochaines exécutions de cette requête. Cette requête n'est pas encore exécutée.

Cette fonction est conseillée lors de l'exécution successive d'une même requête en modifiant uniquement quelques paramètres de cette requête à chaque exécution.

La requête pourra ensuite être exécutée grâce à la fonction `HExécuteRequêteSQL` .

Pour libérer les ressources de cette requête, utilisez la fonction `HAnnuleDéclaration` .

Attention, il est possible que certains fichiers restent ouverts par le moteur SQL de l'AS400 qui anticipe ainsi une prochaine requête.

Les requêtes paramétrées (ou procédures stockées) susceptibles de renvoyer plusieurs résultats ne sont pas gérées.

HExécuteRequêteSQL

Cette fonction exécute une requête préalablement déclarée par `HPrépareRequêteSQL` ou bien sur une chaîne SQL.

La syntaxe avec connexion doit être utilisée si la requête fait référence à un fichier qui n'est pas dans l'analyse ou avec le mode *`HRequêteSansCorrection`*.

Le mode *`HRequêteSansCorrection`* doit être utilisé si la requête fait apparaître un nom de bibliothèque ou des ordres spécifiques.

Paramètres

L'utilisation de paramètres dans la requête (préparée) utilise une syntaxe nommée : chaque nom doit être précédé du caractère ":" (deux points) dans la requête initiale.

C'est en général le moyen le plus performant de passer des paramètres dans une requête.

```
//Préparation de la requête pour de multiples exécutions
HPrépareRequêteSQL ( Insert , Connexion , ...
RequeteSansCorrection , "INSERT INTO PERSONE VALUES (:nom, :prenom,
:age )" )
// Boucle d'exécution de la requête
// Seuls quelques paramètres sont modifiés
POUR i =1 A 10
 Insert.nom = "Nom" + i
 Insert.prenom = "Prenom" + i
 Insert.Age = i
 HExécuteRequêteSQL (Insert)
FIN
```

Exemple

```
////INSERTION BLOB

	HOuvreConnexion(MaConnexionpower8)

	IdLast est un entier

	schaine est une chaîne

	Marequete est une Source de Données

	ASExec("ADDLIBLE CED5")

	xes est un Buffer
	xes = fChargeBuffer("c:\temp\capture.gif")

	schaine="INSERT INTO CEDBLO2(nom,photo) VALUES(:nom,:photo)"

	SI PAS
	HPrépareRequêteSQL(Marequete,MaConnexionpower8,hRequêteSansCorrection,schaine)
	ALORS
		Erreur(HErreurInfo())
	FIN

	Marequete.nom = "aura"
	Marequete.photo = xes

	QUAND EXCEPTION DANS
		SQLTransaction(sqlDébut,MaConnexionpower8)

		SI PAS HExécuteRequêteSQL(Marequete) ALORS
			Info(HErreurInfo())
		FIN
```

```
            SQLTransaction(sqlFin,MaConnexionpower8)

            SI HLitDernier(Cedblo2, Id) ALORS
                  IdLast = Cedblo2.Id
                  Info(IdLast)
            SINON
                  Info("Fichier non trouvé " + ErreurInfo())
             FIN

FAIRE
              SQLTransaction(sqlAnnule, MaConnexionpower8)
FIN
```

SQL sur AS/400

Le langage SQL est extrêmement complet et puissant. La documentation et les ressources sont nombreuses. Un site intéressant, en français, avec beaucoup de ressources iSeries : http://conduitedeprojet.net

Pour concevoir vos requêtes les utilitaires Query/400, Query Manager et SQL de l'AS/400 sont très adaptés, ils se lancent respectivement par les commandes WRKQRY, STRQM et STRSQL.

Le code de la requête pourra ainsi être simplement copié.

Instructions de base

Sélectionner les données
SELECT FROM
SELECT * FROM t1 (toutes les colonnes)

SELECT c1,c2 FROM t1 (sélection des colonnes c1 et c2)
SELECT DISTINCT c1 FROM t1 (élimine les doublons : ne ramène qu'une valeur pour la colonne c1)

SELECT c1 AS "colonne1" FROM t1 (renommer une colonne)

Restreindre la sélection
SELECT * FROM t1 WHERE
SELECT * FROM t1 WHERE c1 IN ('01','02','04')
SELECT * FROM t1 WHERE c2 NOT BETWEEN 10 AND 15
SELECT * FROM t1 WHERE c3 IS NULL
SELECT * FROM t1 WHERE c3 IS NOT NULL
>, >=, <, <=, =, <>, (comparateur arithmétiques)
AND, OR, NOT, (comparateur logique)

% (n'importe quelle séquence de car.)

_ (soulignement) (n'importe quel caractère)

Trier et présenter les résultats
SELECT * FROM t1 ORDER BY c1 (tri ascendant par défaut)
SELECT * FROM t1 ORDER BY c2,c4 (tri par c2 puis tri par c4)
SELECT * FROM t1 ORDER BY c1 ASC, c3 DESC (tri ascendant ou descendant)

Exprimer les jointures

SELECT * FROM t1,t2 (jointure sans qualification = produit cartésien)
SELECT * FROM t1,t2 WHERE t1.c1 = t2.c2 (jointure avec égalité)
SELECT * FROM t1 a,t2 b,t3 c WHERE a.c1=b.c2 AND b.c2=c.c3 (jointures en cascades)

Manipuler les données
SELECT c1,c2*3.25 AS "PRIX" FROM t1
YEAR, MONTH, DATE (date)
SUBSTRING, UPPER, LOWER, CHARACTER_LENGTH (manipulation de chaînes de car.)

Les fonctions statistiques
AVG (moyenne)
COUNT (nombre d'éléments)
MAX (maximum)
MIN (minimum)
SUM (somme)
SELECT COUNT(*)FROM t1
SELECT SUM(c1) FROM t2

Regroupements
SELECT * FROM t1 GROUP BY c1

Sous-requêtes SQL
SELECT * FROM t1 WHERE c1 > (SELECT MIN(c1 FROM t2)
SELECT * FROM t1 WHERE c2 NOT IN (SELECT c2 FROM t2)
SELECT * FROM t1 WHERE c1 > ALL (SELECT c2 FROM t2) (sup. à toutes les valeurs)
SELECT * FROM t1 WHERE c1 > ANY (SELECT c2 FROM t2) (sup. à au moins 1)

Opérateurs ensemblistes

Ils s'intercalent entre deux sélections
UNION (sans les doublons) ou UNION ALL (y compris les doublons)
INTERSECT à partir de la v5r3
EXCEPT à partir de la v5r3

Insérer des enregistrements

INSERT INTO t1 VALUES ('abc',5,7) (toutes les valeurs doivent être renseignées)
INSERT INTO t1(c1,c2) VALUES (1, 'ROUGE') (on ne renseigne que les colonnes indiquées, les colonnes non précisées sont mises à NULL ou à la valeur par défaut si elle est précisée)

INSERT INTO t1 SELECT * FROM t2

Mises à jour d'enregistrement
UPDATE t1 SET c2='ROUGE' WHERE c1=1

Supprimer des enregistrements
DELETE FROM t1 WHERE c1=1
DELETE FROM t1 WHERE c1 IN (SELECT c2 FROM t2)
DELETE FROM t1 (supprime tous les enregistrements de la table t1)

Query et SQL

La commande RTVQMQRY ALWQRYDFN(*YES) permet de convertir un Query vers un membre de fichier source avec une requête SQL et des commentaires.

Jointures

Le CROSS JOIN effectue un produit cartésien entre le contenu de deux tables.

Le INNER JOIN permet de faire une jointure entre deux tables et de ramener les enregistrements de la première table qui ont une correspondance dans la seconde table.

Le LEFT OUTER JOIN entre deux tables retourne tous les enregistrements ramenés par un INNER JOIN plus chaque enregistrement de la table 1 qui n'a pas de correspondance dans la table 2.

Le RIGHT OUTER JOIN entre deux tables retourne tous les enregistrements ramenés par un INNER JOIN plus chaque enregistrement de la table 2 qui n'a pas de correspondance dans la table 1.

Un EXCEPTION JOIN retourne uniquement les enregistrements de la table 1 qui n'on pas de correspondance dans la table 2.

Limiter le nombre de lignes (V5R1)

FETCH FIRST n ROWS ONLY

Exemples

Récupérer le nom du mois en cours

```
MAREQUE est une Source de Données
sChain est une chaîne

sChain="SELECT CASE MONTH(CURRENT DATE) WHEN 1 THEN 'Janvier' WHEN 2
THEN 'Février' WHEN 3 THEN 'Mars' WHEN 4 THEN 'Avril' "
sChain+="WHEN 5 THEN 'Mai' WHEN 6 THEN 'Juin' WHEN 7 THEN 'Juillet'
WHEN 8 THEN 'Août' WHEN 9 THEN 'Septembre' WHEN 10 THEN 'Octobre' "
sChain+="WHEN 11 THEN 'Novembre' ELSE 'Décembre' END AS mois FROM
SYSIBM/SYSDUMMY1 "
HExécuteRequêteSQL(MAREQUE,MaConnexion1,hRequêteSansCorrection,sChai
n)
HLitPremier(MAREQUE)
Info (MAREQUE.mois)
```

Client avec le plus grand nombre de commandes

```
sChain="SELECT CUST_ID , COUNT(*) AS NBRCDE "
sChain+="FROM SP_ORD "
sChain+="GROUP BY CUST_ID "
sChain+="ORDER BY NBRCDE DESC "
sChain+="FETCH FIRST 1 ROWS ONLY"
HExécuteRequêteSQL(mareque,MaConnexion1,hRequêteSansCorrection,sChai
n)
HLitPremier(mareque)
Info(mareque.NBRCDE)

sChain="WITH TMP AS "
sChain+="( "
sChain+="SELECT CUST_ID, COUNT(*) AS NBRCDE "
sChain+="FROM SP_ORD "
sChain+="GROUP BY CUST_ID "
sChain+=") "
sChain+="SELECT * FROM TMP "
sChain+="ORDER BY NBRCDE DESC "
sChain+="FETCH FIRST 1 ROWS ONLY"
HExécuteRequêteSQL(mareque,MaConnexion1,hRequêteSansCorrection,sChai
n)
HLitPremier(mareque)
Info(mareque.NBRCDE)
```

Fonctions W-Langage pour AS/400

Fonctions Appel de Programmes/Procédure

Easycom XML

Introduction

Easycom XML est une interface permettant de décrire et d'appeler les procédures et programmes natifs IBM i. L'échange d'information se fait par XML, les paramètres d'un programme appelé doivent être envoyés au format XML et les résultats de l'appel sont également au format XML.

Les programmes et les structures de données peuvent être décrits en PCML ou en RPG simplifié.

Cette interface permet plus de souplesse dans les appels de programmes. Elle offre également la possibilité d'appeler des services programs ainsi que de recupérer des structures complexes en retour. Elle permet en outre une meilleure gestion des erreurs et de diagnostiquer les problèmes d'appels de programme plus facilement.

Voir les fonctions ASXMLDefinie, ASXMLChargeDefinition, ASXMLAttacheSrvPgm et ASXMLAppelPgm pour plus de détails.

ASXMLDefinie

Charge une définition de procédure ou de programme depuis une description PCML ou RPG simplifié.

Syntaxe

```
Result = ASXMLDefinie(Type, Description, Connexion)
```

En anglais : ASXMLDefine

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

Type

Type de source, les deux seules valeurs acceptées sont PCML ou RPG

Source

Description PCML ou RPG du programme ou de la procédure.

Connexion

Connexion - nom de la connexion

Détail

Les structures de données, les programmes et les procédures sont définis sous forme de chaine dans le code windev, elles sont ensuite stockées dans le job Easycom.

Le RPG simplifié : Le RPG simplifié est un language indentique au RPG, sans les restrictions de colonage, et une instruction se termine par un point-virgule. La directive /COPY peut-être utilisée pour charger des définitions de RPG externe.

Exemple

Appel de la procédure S_FCUST :

```
HOuvreConnexion(MaConnexion1)
sRPG est une chaîne = [
DS_CUST     E DS  extname(S_CUSTOMER);
S_FCUST       PR  LIKEDS(DS_CUST);
TERM1       5P 0
]

SI PAS ASXMLDefinie("RPG",sRPG ,MaConnexion1) ALORS
       Info(ErreurInfo())
SINON
       Info("La definition a été chargée")
```

```
FIN
```

ASXMLChargeDefinition

Charge une définition de procédure ou de programme depuis une description PCML ou RPG simplifié depuis un fichier présent sur l'as400.

Syntaxe

```
Result = ASXMLChargeDefinition(Type, Fichier, Connexion)
```

En anglais : ASXMLLoadDefinition

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

Type

Type de source, les deux seules valeurs acceptées sont PCML ou RPG

Fichier

Chemin complet du fichier sur l'as400.

Connexion

Connexion - nom de la connexion

Détail

Les structures de données, les programmes et les procédures sont définies dans un fichier présent sur l'as400, elles sont chargées et stockées dans le job Easycom.

Le RPG simplifié : Le RPG simplifié est un language indentique au RPG, sans les restrictions de colonage, et une instruction se termine par un point-virgule. La directive /COPY peut-être utilisée pour charger des définitions de RPG externe.

Exemple

```
HOuvreConnexion(MaConnexion1)
sFichier est une chaîne
sFichier = "EASYCOMXMP/QRPGLESRC,CVTNW_H"

SI PAS ASXMLChargeDefinition("RPG",sFichier ,MaConnexion1) ALORS
	Info(ErreurInfo())
SINON
	Info("La definition a été chargée")
FIN
```

ASXMLAttacheSrvPgm

Attache un service programme au job Easycom.

Syntaxe

```
Result = ASXMLAttacheSrvPgm(NomduServiceProgram, Connexion)
```

En anglais : ASXMLBindSrvPgm

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

NomduServiceProgram

Le nom du service program, ce nom peut être qualifié (librairie/nom), si la librairie n'est pas spécifiée, *LIBL est utilisé.

Connexion

Connexion - nom de la connexion

Détail

Attache un service programme au job Easycom. Tous les points d'entrée du service programme sont alors accessibles depuis la fonction ASXMLAppelPgm.

Exemple

```
HOuvreConnexion(MaConnexion1)
sSrvPgm est une chaîne

sSrvPgm = "EASYCOMXMP/XMPSRVPGM"

SI PAS ASXMLAttacheSrvPgm(sSrvPgm ,MaConnexion1) ALORS
       Info(ErreurInfo())
SINON
       Info("Le service programme  a été chargé")
FIN
```

ASXMLAppelPgm

Execute un appel de programme ou de procedure dont le prototype a été préalablement chargé par ASXMLChargeDefinition ou ASXMLDefinie.

Syntaxe

```
Result = ASXMLAppelPgm(Programme,ParametresIN,[Type], Connexion)
```

En anglais : ASXMLCallPgm

Paramètres

Result

Chaine XML- retourne les valeurs de retour du programme sous format XML ou JSON. En cas d'erreur, le xml retourné contient le tag <asxmlerr>. En cas de succès, le xml contiendra les tags <returnvalue> et <parameterlist>.

Programme

Nom du programme à appeler

ParametresIN

Paramètres d'entrée du programme, sous format XML

Type

Type – Type de données, ce paramètre est optionnel, les valeurs autorisées sont JSON ou XML. Si il n'est pas présent le type utilisé est XML.

Connexion

Connexion - nom de la connexion

Détail

L'échange d'informations entre le programme as400 et windev se fait via XML. Les paramètres d'entrée du programme doivent être envoyés dans un format XML correspondant à la structure attendue par le programme.

Exemple de XML en entréé de ASXMLAppelPgm pour l'appel de RPCSAMPLE :

```
<?xml version="1.0"  encoding="UTF-8" ?>
<Easycom Version="02.02.004">
 <Program Name="RPCSAMPLE">
  <ParameterList>
    <OP1>2</OP1>
    <STR1>aura</STR1>
    <OP2>3</OP2>
    <STR2>test</STR2>
    <OP3>4</OP3>
 </ParameterList>
 </Program>
</Easycom>
```

Les valeurs de retour sont également au format XML, à l'aide du tag particulier **<PARAMETERLIST>** :

```
<?xml version="1.0"  encoding="UTF-8" ?>
<Easycom Version="2.01">
<Program Name="RPCSAMPLE">
  <ParameterList>
    <OP2 Type="Packed">5.00</OP2>
    <STR2 Type="Char">aura</STR2>
    <OP3 Type="Packed">6.0000</OP3>
  </ParameterList>
 </Program>
```

Exemple complet

Appel du programme RPCSAMPLE dont la structure est définie en pcml.

Remarque : dans le pcml, pour la variable path, LIB et PGM doivent être en minuscules :

path="/QSYS.lib/EASYCOMXMP.lib/RPCSAMPLE.pgm"

```
sPCML est une chaîne
sParam est une chaîne
sResultat est un chaîne ANSI


sPCML = [
       <pcml version="4.0">
       <program name="RPCSAMPLE" path="/QSYS.lib/EASYCOMXMP.lib/RPCSAMPLE.pgm" >
       <data name="OP1" type="packed" length="5" precision="2" usage="input" />
       <data name="STR1" type="char" length="20" usage="input" />
       <data name="OP2" type="packed" length="5" precision="2" usage="inputoutput" />
       <data name="STR2" type="char" length="30" usage="inputoutput" />
       <data name="OP3" type="packed" length="10" precision="4" usage="output" />
       </program>
       </pcml>
```

```
]

sParam = [
       <OP1>%1</OP1>
       <STR1>%2</STR1>
       <OP2>%3</OP2>
       <STR2>%4</STR2>
       <OP3>%5</OP3>
]

STStrpcsample est Structure
       OP1 est réel
       STR1 est chaîne sur 20
       OP2 est réel
       STR2 est chaîne sur 30
       OP3 est réel
FIN


stRpcSample est STStrpcsample
stRpcSample.OP1 = 2
stRpcSample.STR1="aura"
stRpcSample.OP2 = 3
stRpcSample.STR2 = "test"
stRpcSample.OP3 = 4

sParam =
ChaîneConstruit(sParam,stRpcSample.OP1,stRpcSample.STR1,stRpcSample.OP2,stRpcSample.S
TR2,stRpcSample.OP3)

SI PAS ASXMLDefinie("PCML",sPCML,PrinciConnexion) ALORS
       Info(ErreurInfo())
SINON

       sResultat = ASXMLAppelPgm("RPCSAMPLE",sParam,PrinciConnexion)

       Désérialise(stRpcSample,sResultat,psdXMLAgrégé)
       Info("OP1="+ stRpcSample.OP1 + " - STR1=" + stRpcSample.STR1 + " - OP2=" +
stRpcSample.OP2 + " - STR2=" + stRpcSample.STR2 + " - STR3=" + stRpcSample.OP3)
       Info(sResultat)
FIN
```

AsAppelRTV

Lance l'exécution de commandes AS/400 ayant des variables de retour (commandes retrieve).

Syntaxe

```
Result = ASAppelRtv( Commande [,Connexion])
```

En anglais : `ASRtvCall`

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

Commande

Chaîne - le détail de la commande à exécuter

Connexion [optionnel]

Connexion - nom de la connexion

Détail

L'utilisation des commandes retrieve implique un niveau de connaissance minimum de la programmation AS/400. Le plus simple étant bien sûr de composer la commande au préalable sur un terminal.

Les principales commandes AS/400 renvoyant des variables sont les commandes retrieve (**RTV***) et certaines commandes receive (**RCV***).

Il est possible d'exécuter les commandes supplémentaires développées spécifiquement.

Ces commandes peuvent retourner une ou plusieurs variable(s) passée(s) en paramètre(s).

Ces variables n'ont pas besoin d'être déclarées si elles font partie des variables prédéfinies de la fonction, en revanche l'appel d'une commande où une variable peut ne pas être connue doit s'accompagner d'une déclaration (par exemple un le RTNVAR pour un RTVDTAARA).

Dans ce cas la syntaxe est de la forme (les indications entre [et] sont optionnelles)

[NomVar1=Type;[NomVar2=Type;...]]NomCommande paramètre1(NomVar1) paramètre2(NomVar2) ...]

`NomVarn` est un nom de variable

`Type` est un type AS/400 (identique à un source CL)

`NomCommande` est le nom de la commande à exécuter

`Paramètren` est un paramètre de la commande (identique à un source CL)

La récupération du résultat d'une commande retrieve se fait par la fonction ASRésultatRtv.

Cette commande renvoie toujours vrai, il faut ensuite tester la variable réservée "RC".

Exemple

```
LigneCmd est une chaîne
LigneCmd="CCSID=DEC(5 0);RTVJOBA JOB(&JOB) USER(&USER) USRLIBL(&USRLIB)
SYSLIBL(&SYSLIB) CCSID(&CCSID) CURLIB(&CURL)"

HOuvreConnexion(MaConnexionpower8)

SI PAS ASAppelRtv(LigneCmd)ALORS
        Info(ErreurInfo(errRésumé))
FIN

sResultat est une chaîne
sResultat = ASResultatRtv("RC")
SI sResultat = "0" ALORS
        Info(ASResultatRtv("CCSID"))
SINON
        Info("Erreur de la commande : "+sResultat)
FIN
```

AsResultatRTV

Cette fonction s'utilise toujours après la fonction AsAppelRTV, elle récupère les variables de la commande de type RTV ou RCV.

Syntaxe

```
Resultat = ASResultatRtv(NomVariable [,Connexion])
```

En anglais : `ASRtvResult`

Paramètres

Resultat

Chaîne contenant le résultat de la récupération

NomVariable

Chaîne de caractères contenant le nom de la variable à lire.

Connexion (optionnel)

Connexion - Nom de la connexion

Détail

ASResultatRtv permet de récupérer sous forme d'une chaîne de caractères la valeur d'une ou de toutes les variables déclarées lors de l'appel à la commande retrieve.

L'exécution d'une commande retrieve est réalisée par la fonction ASAppelRTV.

RC est un nom de variable réservé : si la syntaxe est correcte mais si l'exécution échoue, la fonction ne provoque pas une erreur mais va mettre à jour une variable spéciale dont le nom est RC. Cette variable contient 0 en cas de succès et le code CPF provoqué en cas d'échec. Il s'agit donc de vérifier systématiquement que la variable RC est bien nulle avant de récupérer les variables propres à la commande.

Exemples

1) Récupérer les propriétés du job

```
Lignecmd est une chaîne
LigneCmd = "RTVJOBA USER(&USER) USRLIBL(&USRLIB) SYSLIBL(&SYSLIB) CURLIB(&CURL)"

ASAppelRtv(lignecmd)

// Récupération des paramètres
sResultat est une chaîne
sResultat = ASResultatRtv("RC")
SI sResultat = "0" ALORS
        sUser=ASResultatRtv("user")
        sUserlib=ASResultatRtv("usrlib")
        sSyslib=ASResultatRtv("syslib")
        sCurl=ASResultatRtv("curl")
FIN
```

2) Récupérer le numéro de série

```
bRet est un booléen
Resultat est une chaîne
var1 est une chaîne
LigneCmd est une chaîne

lignecmd = "RTVSYSVAL SYSVAL(QSRLNBR) RTNVAR(&VAR1)"
bRet = ASAppelRtv(lignecmd)

// Lecture du résultat
```

```
sResultat est une chaîne
sResultat = ASResultatRtv("RC")
SI sResultat = "0" ALORS
        Resultat = ASResultatRtv("VAR1")
        Info("Numéro de série de l'AS/400 : " + Resultat)
FIN
```

AsExec

Lance l'exécution d'une commande sur AS/400.

Pour appeler une commande de type Retrieve voir les fonctions ASAppelRTV et ASRésultatRTV.

Pour appeler un programme, voir la fonction ASLanceRPC.

Syntaxe

```
Result = ASExec(Commande [, Connexion])
```

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

Commande

Chaîne - le détail de la commande à exécuter

Connexion [optionnel]

Connexion - nom de la connexion

Détail

Seules les commandes **NON INTERACTIVES** (pas d'entrées clavier et de sorties écran) de l'AS/400 peuvent êtres lancées. Cette fonction ne permet pas de récupérer les valeurs de retour de la commande lancée (voir ASAppelRTV et ASRésultatRTV).

La fonction **ASExec** permet d'envoyer toutes sortes de commandes et elle peut être très utile à la gestion des bibliothèques (ADDLIBLE, CHGCURLIB...) et des fichiers (OVRDBF...).

Il est conseillé d'utiliser ASExec lorsque le programme ne retourne pas de valeurs.

ASLanceRPC est à utiliser dans tous les autres cas.

Exemples

// Envoi du message "Bonjour" à l'utilisateur "QPGMR"

```
ASExec("SNDMSG MSG('BONJOUR')TOUSR(QPGMR)")
```

// Changement de la bibliothèque courante

```
ASExec("CHGCURLIB PROD2005")
```

// Appeler un programme

```
ASExec("CALL PGM(MYPROG) PARM('00213')")
```

// Création d'un récepteur, d'un journal et démarrage de la journalisation avec contrôle des erreurs

```
sCmd est une chaîne
sCmd="CRTJRNRCV JRNRCV(EASYCOM/TMPRCV)"
SI PAS ASExec(sCmd) ALORS
```

```
 SI ExtraitChaîne(ErreurInfo(),2,CR)="CPF7010" ALORS Info("Récepteur déjà présent") SINON
Info(ErreurInfo)
FIN

sCmd="CRTJRN JRN(EASYCOM/TMPJRN) JRNRCV(EASYCOM/TMPRCV)"
SI PAS ASExec(sCmd) ALORS
 SI ExtraitChaîne(ErreurInfo(),2,CR)="CPF7015" ALORS Info("Le récepteur contient déjà ce poste
de journal") SINON Info(ErreurInfo)
FIN

sCmd="STRJRNPF FILE(EASYCOM/SP_CUST) JRN(EASYCOM/TMPJRN)"
SI PAS ASExec(sCmd) ALORS
 SI ExtraitChaîne(ErreurInfo(),2,CR)="CPF7030" ALORS Info("Fichier déjà journalisé") SINON
Info(ErreurInfo)
FIN
```

Dans l'exemple avec AsAppelRTV et ASRésultatRTV on utilise un RTVMBRD pour récupérer des propriétés d'un fichier mais on peut imaginer aussi la création d'un fichier avec un DSPFD

```
ASExec("DSPFD FILE(EASYCOM/SP_CUST) TYPE(*MBR) OUTPUT(*OUTFILE) OUTFILE(QTEMP/SORTIE)")
sortie est une Source de Données
HDéclareExterne("QTEMP/SORTIE",sortie,MaConnexion)
HLitPremier(sortie)
Info("Nombre d'enregistrements :"+sortie.mbnrcd)
Info("Nombre d'enregistrements supprimés :"+sortie.mbndtr)

ASExec("DLTF FILE(QTEMP/SORTIE)")
```

AsLanceRPC

Lance l'exécution d'un programme AS/400.

ASLanceRPC ne concerne que les appels de programmes avec des paramètres.

Pour appeler un programme sans paramètres (ou seulement des paramètres en entrée), voir la fonction ASExec.

Pour appeler une commande de type Retrieve comme RTVJOBA, voir les fonctions ASAppelRTV et ASRésultatRTV.

Pour pouvoir être utilisé par ASLanceRPC, le programme doit avoir été décrit et importé dans l'analyse. Le nombre maximum de paramètres est 200.

On peut aussi paramétrer un temps d'exécution maximum (timeout) pour les appels de programmes depuis la configuration EASYCOM Serveur.

Syntaxe

```
Result = ASLanceRPC(NomProgramme)

En anglais : ASRunRPC
```

Paramètres

Result

Booléen - Vrai si l'appel au programme a réussi, Faux en cas d'erreur.

NomProgramme

Chaîne - le nom du programme correspondant au fichier importé dans l'analyse.

Remarque : cette fonction utilise nécessairement la connexion associée au programme (fichier) dans l'analyse.

Détail

Pour être exploitable par la fonction **ASLanceRPC,** le programme AS/400 doit avoir une procédure de description EASYCOM sur l'AS/400 (description faite avec le constructeur RPC-DTAQ) et être importé dans l'analyse du projet WinDev, sous forme de fichier.

Avant exécution, il faut initialiser les paramètres en entrée (In) et en entrée/sortie (In/Out) dans les rubriques correspondantes du buffer du programme, comme n'importe quel fichier.

Après exécution, les paramètres en sortie (Out) et en entrée/sortie (In/Out) sont stockés dans les rubriques correspondantes du buffer du fichier Hyper File.

Avec des champs de saisie liés au programme (fichier), le code suivant va passer les valeurs saisies au programme et afficher les éventuelles valeurs de retour :

```
EcranVersFichier()
ASLanceRPC(RPCSAMPLE)
FichierVersEcran()
```

Notes

L'exécution du programme est réalisée dans la session de la connexion associée au programme. Les commandes CL lancées par **ASExec**, et les programmes appelés par **ASLanceRPC** sont exécutés dans l'environnement du job (voir aussi Job EASYCOM).

Si vous appelez un programme, sans paramètres de retour, vous pouvez utiliser ASExce :

```
ASExec("CALL PGM(MyProgram) PARM('My param 1' 'My Param 2')")
```

Exemple

La description du programme RPCSAMPLE a été importé dans l'analyse.

```
RPCSAMPLE.OP1=2
RPCSAMPLE.OP2=3
RPCSAMPLE.OP3=0
RPCSAMPLE.STR1="aura"
RPCSAMPLE.STR2=" "

SI ASLanceRPC(RPCSAMPLE) ALORS
        Info("OP1=" + RPCSAMPLE.OP1)
        Info("OP2=" + RPCSAMPLE.OP2)
        Info("OP3=" + RPCSAMPLE.OP3)
        Info("STR1=" + RPCSAMPLE.STR1)
        Info("STR2=" + RPCSAMPLE.STR2)
SINON
        Info(HErreurInfo())
FIN
```

Voir l'exemple Appel de programmes.

ASAppelPgm

(en anglais `ASPgmCall`)

ASAppelPgm est une alternative à ASLanceRPC.

Le but est le même : l'appel de programmes natifs, mais avec une autre approche.

Tout d'abord, il n'est pas nécessaire d'importer la description du programme dans l'analyse.

Cette description pourra être :

- soit en syntaxe PCML intégrée au programme
- soit en syntaxe PCML dans un fichier de l'IFS

- soit une description de programme effectuée avec le configurateur RPC/DATAQ.

Ensuite, les valeurs d'entrée/sortie sont passées directement à la fonction, soit un par un, soit par une variable de type structure.

Syntaxe

```
Result = ASAppelPgm(DescProgramme, NomProgramme [, Connexion], Param1 [, Param2 [, Parm3, ...] ])
```

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

DescProgramme

Chaîne – correspond à l'identité du programme. Peut être :

- *PGM/PGMNAME si PGMNAME est décrit via le configurateur RPC/DTAQ

- un chemin de l'IFS (par exemple "/pgm_defs/pgmname.pcml") vers un fichier PCML contenant la description

- une chaîne multi-ligne contenant le source PCML correspondant à la description du programme.

NomProgramme

Chaîne – correspond au nom réel du programme, ou chaîne vide si on doit utiliser le nom présent via DescProgramme. Est **ignoré** si DescProgramme est de la forme *PGM/XXX.

Connexion [optionnel]

Connexion - nom de la connexion

Param1, Parm2 , …

Variable ou valeurs correspondant aux paramètres.

Détail

Pour une première approche, vous pouvez consulter l'exemple 'PgmCall' qui montre différentes façons d'appeler le programme RPCSAMPLE via ASAppelPgm.

Pour les programmes simples, on peut par exemple donner les paramètres un par un à ASAppelPgm. Par exemple :

```
SI PAS ASPgmCall("*PGM/RPCSAMPLE", "", cnx, OP1, STR1, OP2, STR2, OP3) ALORS
 Erreur(ErreurInfo(errComplet))
FIN
```

Il faut dans ce cas que les valeurs fournies soient bien sûr **dans le même ordre**. La description fournie (ici une description Easycom) sert à connaître les tailles AS/400 et usages de chaque élément.

Il est également possible de fournir une structure. Par exemple :

```
STStrpcsample est structure
 rOp1 est réel
 sStr1 est chaîne sur 20
 sOp2 est réel
 sStr2 est chaîne sur 30
 rOp3 est réel
FIN
stRpcSample est STStrpcsample


stRpcSample:rOp1 = OP1
stRpcSample:sStr1 = STR1
stRpcSample:sOp2 = OP2
```

```
stRpcSample:sStr2 = STR2

SI PAS ASPgmCall("*PGM/RPCSAMPLE", "", cnx, stRpcSample) ALORS
 Erreur(ErreurInfo(errComplet))
FIN
```

La structure devra bien entendu refléter les paramètres véritables.

Enfin, il est également possible d'utiliser une syntaxe PCML présente dans un fichier de l'IFS. Par exemple:

```
SI PAS ASPgmCall("/tmp/rpcsample.pcml", "", cnx, OP1, STR1, OP2, STR2, OP3) ALORS
 Erreur(ErreurInfo(errComplet))
FIN
```

La syntaxe PCML pour RPCSAMPLE est la suivante :

```
<pcml version="1.0">
<!-- PCML source for calling "RPCSAMPLE" program -->
<!-- Program "RPCSAMPLE" and its parameter list -->

<program name="RPCSAMPLE" path="/*LIBL.lib/RPCSAMPLE.pgm">

<data name="Op1" type= length="5" precision="2" usage="input"/>
<data name="Str1" type="char" length= usage="input"/>
<data name="Op2" type="packed" length="5" precision="2" usage="inputoutput"/>
<data name="Str2" type="char" length="30" usage="inputoutput"/>
<data name="Op3" type="packed" length="10" precision="4" usage="output"/>

</program>

</pcml>
```

Il est également possible de passer directement le PCML à la place du nom du fichier.

C'est un langage conçu par IBM pour les appels de programmes et peut être généré en sortie de compilation COBOL ou RPG ILE. Nous y retrouvons les types natifs, et la possibilité de decrire une procédure de programme de service.

Exemple 1

Dans cet exemple, on appelle le programme RPCSAMPLE (qui a été défini avec le configurateur RPC/DataQueue) en utilisant une structure pour les paramètres :

```
STStrpcsample est Structure
     rOp1 est réel
     sStr1 est chaîne sur 20
     sOp2 est réel
     sStr2 est chaîne sur 30
     rOp3 est réel
FIN
stRpcSample est STStrpcsample


stRpcSample:rOp1 = 2
stRpcSample:sStr1 = "aura"
stRpcSample:sOp2 = 3
stRpcSample:sStr2 = "test"
stRpcSample:rOp3 = 4

HOuvreConnexion(MaConnexionpower8)

SI PAS ASPgmCall("*PGM/RPCSAMPLE", "", MaConnexionpower8, stRpcSample) ALORS
     Info(ErreurInfo())
SINON
```

```
        Info("OP1="+ stRpcSample.rOp1 + " - STR1=" + stRpcSample.sStr1 + " - OP2=" +
        stRpcSample.sOp2 + " - STR2=" + stRpcSample.sStr2 + " - STR3=" +
        stRpcSample.rOp3)
FIN
```

Exemple 2

Dans cet exemple, on appelle le programme RPCSAMPLE en utilisant une définition PCML dans le code :

```
sPCML est une chaîne


HOuvreConnexion(MaConnexionpower8)

sPCML = [
      <pcml version="4.0">
      <program name="RPCSAMPLE" path="/QSYS.lib/EASYCOMXMP.lib/RPCSAMPLE.pgm" >
      <data name="OP1" type="packed" length="5" precision="2" usage="input" />
      <data name="STR1" type="char" length="20" usage="input" />
      <data name="OP2" type="packed" length="5" precision="2" usage="inputoutput" />
      <data name="STR2" type="char" length="30" usage="inputoutput" />
      <data name="OP3" type="packed" length="10" precision="4" usage="output" />
      </program>
      </pcml>
]


STStrpcsample est Structure
      OP1 est réel
      STR1 est chaîne sur 20
      OP2 est réel
      STR2 est chaîne sur 30
      OP3 est réel
FIN

stRpcSample est STStrpcsample
stRpcSample.OP1 = 2
stRpcSample.STR1="aura"
stRpcSample.OP2 = 3
stRpcSample.STR2 = "test"
stRpcSample.OP3 = 4


SI PAS ASPgmCall(sPCML, "", MaConnexionpower8, stRpcSample) ALORS
      Info(ErreurInfo())
SINON
      Info("OP1="+ stRpcSample.OP1 + " - STR1=" + stRpcSample.STR1 + " - OP2=" +
stRpcSample.OP2 + " - STR2=" + stRpcSample.STR2 + " - STR3=" + stRpcSample.OP3)

FIN
```

Exemple 3

Dans cet exemple, on appelle le programme le programme RPG suivant utilisant des structures imbriquées :

```
D DS_A          DS
D  MBR1                  10a
D  MBR2                   8p 2
D  MBR3                   8p 2

D DS_P          DS              qualified
D  MBRD                         likeds(ds_A)

D  PARM1        DS              likeds(ds_P)

D   PARMT       S          10a

D   PARMA       S           8p 2   Dim(3)
D   PARMI       S          8p 2


C    *entry      plist
C              PARM              PARM1
C              PARM              PARMT
C              PARM              PARMA
C              PARM              PARMI

 /free

   PARM1.MBRD.MBR2 = PARM1.MBRD.MBR3;
   PARMT = 'azeOK';

   PARMA(1) = 1 + PARMA(1);

    PARMA(2) = 2 + PARMA(2);
    PARMA(3) = 3 + PARMA(3);

    PARMI = PARMI + 1;
    return;

  /end-free
```

Depuis Windev 27, on peut envoyer directement une structure Windev complexe (structures Windev contenant d'autres structures Windev), représentant donc fidèlement les paramètres de leurs programmes (description structure et/ou programme d'un PCML) même les plus compliquées, et ce sans faire de mapping, de découpage, en une seule variable qui sera remplie/mise à jour directement par Easycom.

```
sPCML est une chaîne
sPCML = [
<pcml version="6.0">
    <struct name="DS_A">
        <data name="MBR1" type="char" length="10" usage="inherit" />
        <data name="MBR2" type="packed" length="8" precision="2" usage="inherit" />
        <data name="MBR3" type="packed" length="8" precision="2" usage="inherit" />
    </struct>
    <!-- 7 -->
    <struct name="DS_P">
```

```
        <data name="MBRD" type="struct" struct="DS_A" usage="inherit" />
    </struct>
    <program name="SAMPLEDS6" entrypoint="SAMPLEDS6">
        <data name="PARM1" type="struct" struct="DS_P" usage="inputoutput" />
        <data name="PARMT" type="char" length="10" usage="inputoutput" />
        <data name="PARMA" type="packed" length="8" precision="2"
usage="inputoutput" count="3" />
        <data name="PARMI" type="packed" length="8" precision="2"
usage="inputoutput" />
    </program>
</pcml>
]

DS_A est Structure
    mbr1    est une chaîne sur 10
    mbr2    est un réel
    mbr3    est un réel
FIN
sDS_A    est DS_A

DS_P    est Structure
    mbrd est une DS_A
FIN
sDS_P est DS_P

sDS_A:mbr1    = "aura"
sDS_A:mbr2    = 1
sDS_A:mbr3    = 3

sDS_P.mbrd    = sDS_A
PARMA est un tableau de 3 réels
PARMA[1] = 10
PARMA[2] = 20
PARMA[3] = 30
PARMT est une chaîne
PARMT = "qsd"
PARMI est un réel
PARMI = 2;

completeStruc est une Structure
    PARM1 est DS_P
    PARMT est une chaîne
    PARMA est un tableau de 3 réels
    PARMI est un réel
FIN

SAMPLEDS6 est completeStruc
SAMPLEDS6.PARM1 = sDS_P
SAMPLEDS6.PARMT = PARMT
SAMPLEDS6.PARMA = PARMA
SAMPLEDS6.PARMI = PARMI


HOuvreConnexion(MaConnexion)

ASExec("ADDLIBLE EASYCOMXM3",MaConnexion)

// Nouvelle façon, compatible avec les structures complexes et imbriquées

SI PAS ASPgmCall(sPCML, "SAMPLEDS6",MaConnexion, SAMPLEDS6) ALORS
    Info(ErreurInfo(errComplet))
    Info("PARM1.MBRD.MBR2=" + sDS_P:mbrd:mbr2)
```

```
SINON
    Trace("PARM1.MBRD.MBR2=" + SAMPLEDS6:PARM1:mbrd:mbr2)
    Trace("PARM1.MBRD.MBR3=" + SAMPLEDS6:PARM1:mbrd:mbr3)
    Trace("PARMT=" + SAMPLEDS6:PARMT)
    Trace("PARMA[1]=" + SAMPLEDS6:PARMA[1])
    Trace("PARMA[2]=" + SAMPLEDS6:PARMA[2])
    Trace("PARMA[3]=" + SAMPLEDS6:PARMA[3])
    Trace("PARMI=" + SAMPLEDS6:PARMI)
FIN
```

Avant Windev 26, l'appel du programme se faisait ainsi :

```
SI PAS ASPgmCall(sPCML, "SAMPLEDS6",MaConnexion, sDS_P, PARMT, PARMA, PARMI) ALORS
    Info(ErreurInfo(errComplet))
    Info("PARM1.MBRD.MBR2=" + sDS_P:mbrd:mbr2)
SINON
    Trace("sDS_P.MBRD.MBR2=" + sDS_P:mbrd:mbr2)
    Trace("sDS_P.MBRD.MBR3=" + sDS_P:mbrd:mbr3)
    Trace("PARMT=" + PARMT)
    Trace("PARMA[1]=" + PARMA[1])
    Trace("PARMA[2]=" + PARMA[2])
    Trace("PARMA[3]=" + PARMA[3])
    Trace("PARMI=" + PARMI)
FIN
```

ASAppelProcedure

ASAppelProcedure est une alternative à ASLanceRPC, pour appeler une procédure de programmes de service (*SRVPGM) ILE.

Le fonctionnement est identique à ASAppelPgm. L'unique différence réside sur le fait de pouvoir désigner un nom de procédure.

Voir ASAppelPgm pour plus d'informations

Syntaxe

```
Result = ASAppelProcedure(DescProcédure, NomServiceProgramme, NomProcédure [,
Connexion], Param1 [, Param2 [, Parm3, ...] ])
```

En Anglais : ASProcedureCall

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

DescProcédure

Chaîne – correspond à l'identité de la procédure/programme. Peut être :

- *PGM/PGMNAME si PGMNAME est décrit via le configurateur RPC/DTAQ

- un chemin de l'IFS (par exemple "/pgm_defs/pgmname.pcml") vers un fichier PCML contenant la description

- une chaîne multi-ligne contenant le source PCML correspondant à la description du programme.

NomduServiceProgram

Le nom du service program, ce nom peut être qualifié (librairie/nom). Si la librairie n'est pas spécifiée, *LIBL est utilisé.

NomProcédure

Chaîne – correspond au nom réel de la procédure, ou chaîne vide si on doit utiliser le nom présent via DescProcédure. Est **ignoré** si DescProcédure est de la forme *PGM/XXX.

Connexion [optionnel]

Connexion - nom de la connexion

Param1, Parm2 , …

Variable ou valeurs correspondant aux paramètres.

Exemples

Exemple 1 :

Dans cet exemple, on appelle le programme RPCSAMPLE en utilisant une structure pour les paramètres :

```
STStrpcsample est Structure
	rOp1 est réel
	sStr1 est chaîne sur 20
	sOp2 est réel
	sStr2 est chaîne sur 30
	rOp3 est réel
FIN
stRpcSample est STStrpcsample


stRpcSample:rOp1 = 2
stRpcSample:sStr1 = "aura"
stRpcSample:sOp2 = 3
stRpcSample:sStr2 = "test"
stRpcSample:rOp3 = 4

HOuvreConnexion(MaConnexionpower8)

SI PAS ASAppelProcedure("*PGM/RPCSAMPLE", "", "",MaConnexionpower8, stRpcSample)
ALORS
	Info(ErreurInfo())
SINON
	Info("OP1="+ stRpcSample.rOp1 + " - STR1=" + stRpcSample.sStr1 + " - OP2=" +
	stRpcSample.sOp2 + " - STR2=" + stRpcSample.sStr2 + " – OP3=" +
	stRpcSample.rOp3)
FIN
```

Exemple 2 :

Dans l'exemple qui suit, on appelle le programme RPCSAMPLE en utilisant une structure pour les paramètres, et en spécifiant un programme de service :

```
STStrpcsample est Structure
	rOp1 est réel
	sStr1 est chaîne sur 20
	sOp2 est réel
	sStr2 est chaîne sur 30
	rOp3 est réel
```

```
FIN
stRpcSample est STStrpcsample


stRpcSample:rOp1 = 2
stRpcSample:sStr1 = "aura"
stRpcSample:sOp2 = 3
stRpcSample:sStr2 = "test"
stRpcSample:rOp3 = 4

sSrvPgm est une chaîne

sSrvPgm = "EASYCOMXMP/XMPSRVPGM"

HOuvreConnexion(MaConnexionpower8)

SI PAS ASAppelProcedure("*PGM/RPCSAMPLE", sSrvPgm, "",MaConnexionpower8, stRpcSample)
ALORS
       Info(ErreurInfo())
SINON
       Info("OP1="+ stRpcSample.rOp1 + " - STR1=" + stRpcSample.sStr1 + " - OP2=" +
stRpcSample.sOp2 + " - STR2=" + stRpcSample.sStr2 + " - STR3=" + stRpcSample.rOp3)

FIN
```

Fonctions IFS

ASfChargeTexte - ASfChargeBinaire

Ces fonctions permettent de charger dans une variable le contenu d'un fichier de l'IFS.

Syntaxe

```
sDonnees = ASfChargeTexte(NomFichier [,Connexion])
sDonnees = ASfChargeBinaire(NomFichier [,Connexion])
```

En anglais : `ASfLoadText, ASfLoadBinary`

Paramètres

NomFichier

Chaîne de caractères contenant le nom du fichier à lire.

sDonnees

Chaîne texte ou binaire, buffer contenant le fichier complet.

Connexion (optionnel)

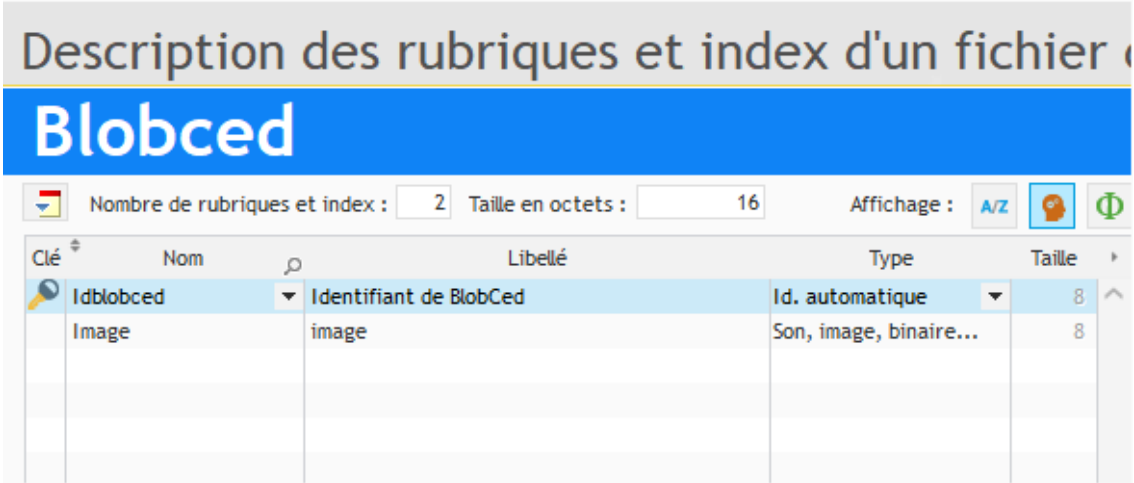
Connexion - Nom de la connexion.

Détail

ASfChargeFichierTexte/Binaire réalisent la lecture complète d'un fichier de l'IFS vers une variable. La différence entre les deux méthodes tient à la conversion ou non vers le jeu de caractères local.

Exemple

Dans cet exemple, la table suivante a été créée dans l'analyse :



Elle a été exportée vers l'AS400.

La table de l'analyse a ensuite été effacée, et la table AS400 réimportée.

Une image présente sur l'IFS est chargée dans un buffer avec la fonction `ASfChargeBinaire`, et ensuite un nouvel enregistrement est inséré dans la table avec la fonction `HAjoute.`

Le dernier enregistrement est ensuite lu et l'image sauvée sur le PC avec la fonction `HExtraitMémo.`

```
id est un entier
bufIn est un Buffer

HOuvreConnexion(MaConnexionpower8)

QUAND EXCEPTION DANS
        zsChemin est une chaîne
        zsChemin = "/tmp/Photo.gif"
        bufIn = ASfChargeBinaire(zsChemin,MaConnexionpower8)

        Blobced.Image = bufIn

        SI HAjoute(Blobced) ALORS
             Info("ajoute done!")
           SINON
                   Info(ErreurInfo())
           FIN


        // récupération de la valeur de l'ID automatique du dernier
        enregistrement
           SI HLitDernier(Blobced, Idblobced) ALORS
               Info(Blobced.Idblobced)
               id = Blobced.Idblobced

           SINON
                    Info("Fichier non trouvé " + ErreurInfo())
           FIN
```

```
FAIRE
	Info("exception!")
	Info(ExceptionInfo())
FIN

////////////////////////////////////////
// Lecture de l'enregistrement
////////////////////////////////////////
 QUAND EXCEPTION DANS
	Info(id)
	HRAZ(Blobced,Idblobced)
	SI HLitRechercheDernier(Blobced,Idblobced,id)
		zsCheminc est une chaîne
		zsCheminc = "C:\temp\Photo6.gif"
			// Extrait le document sur le disque
			SI	HExtraitMémo(Blobced, Image, zsCheminc) ALORS
				Info("extrait!")
			SINON
				Info(ErreurInfo())
			FIN
	   SINON
			Info("Fichier non trouvé " + ErreurInfo())
		FIN
FAIRE
		Info("exception!")
		Info(ExceptionInfo())
FIN

HFermeConnexion(MaConnexionpower8)
```

ASfCrée

Cette fonction permet l'ouverture en création d'un fichier de l'IFS. (ASfOuvre le permet également en utilisant le paramètre Mode).

Syntaxe

```
Résultat = ASfCrée(NomFichier, [,mode] [,Connexion])
```

En anglais : `ASfCreate`

Paramètres

Résultat : entier

Correspond au numéro d'ouverture du fichier. Cet entier doit être ensuite fourni aux fonctions ASfFerme, ASfLit, ASfEcrit, etc.

Le résultat est -1 en cas d'erreur.

Mode : Entier optionnel

Mode de création du fichier.
La valeur par défaut est :
ASfoCréation+*ASfoTronque*+*ASfoEcriture*+*ASfoDroitsU_RW*+*ASfoDroitsG_RW*

Connexion : nom ou variable Connexion optionnelle

Nom de la connexion associée à la sauvegarde.

Exemple

```
nres = ASfCree("/texte/texte_unicode_a_creer.txt",ASfoUnicode+ASfoTronque)
IF nres = -1 ALORS
 Erreur(ErreurInfo())
 RETOUR
END
```

Voir exemple complet.

ASfEcrit

Ecriture dans un fichier de l'IFS.

Syntaxe 1

```
Résultat = ASfEcrit(id_ouverture, donnnes1, taille)
```

Syntaxe 2

```
Résultat = ASfEcrit(id_ouverture, donnees2)
```

En anglais : `ASfWrite`

Paramètres

Résultat : entier

Nombre d'octets ou de caractères écrits.

id_ouverture : Entier

Entier fourni par la fonction ASfCree ou ASfOuvre.

données1 : Buffer ou chaine de caractères

Espace de données à écrire, correspondant à la taille fournie.

données2 : Chaine de caractères

Chaîne de caractère à écrire.

taille

Dans la syntaxe 1, taille du buffer ou de la chaîne à écrire. S'il s'agit d'une chaîne Unicode, <taille> doit être égal au nombre de caractères.

Voir exemple complet.

ASfEcritLigne

Ecriture d'une ligne dans un fichier IFS.

Syntaxe

```
Résultat = ASfEcritLigne(id_ouverture, texte [,taille])
```

En anglais : `ASfWriteLine`

Paramètres

Résultat : entier
Nombre de caractère écrits.
id_ouverture : Entier
Entier fourni par la fonction ASfCree ou ASfOuvre.
texte : chaine Unicode ou ANSI
Texte à insérer.
taille : entier (optionnel).
Nombre de caractères à écrire. Peut être utile si le caractère 0 est présent.

Voir exemple complet.

ASfFerme

Cette fonction ferme le fichier IFS. L'écriture différée éventuelle sera définitive à partir de ce point.

Syntaxe

```
Résultat = ASfFerme(id_ouverture)
```

En anglais : `ASfClose`

Paramètres

Résultat : booléen
Le résultat est égal à faux en cas d'erreur. (Par exemple id_ouverture non valide).
id_ouverture : Entier
Entier fourni par la fonction ASfCree ou ASfOuvre.

Voir exemple complet.

ASfLit

Lecture d'un fichier de l'IFS.

Syntaxe 1 (conseillée)

```
Résultat1 = ASfLit(id_ouverture, taille, données)
```

Syntaxe 2

```
Résultat2 = ASfLit(id_ouverture, taille)
```

En anglais : `ASfRead`

Paramètres

Résultat1 : entier

Nombre d'octets ou de caractères lus (suivant le mode d'ouverture).

Résultat2 : Buffer

Données lues.

id_ouverture : Entier

Entier fourni par la fonction ASfCree ou ASfOuvre.

données : Buffer ou chaine de caractères allouée

Espace de données destiné à recevoir la lecture.

taille : Entier optionnel

Nombre d'octets ou de caractères à lire.

La syntaxe 1 est avantageuse car convertit dans le type fourni (depuis ou vers Unicode si nécessaire), indépendamment du type d'ouverture.

Le nombre donné en paramètre et en résultat correspond à la taille « naturelle », soit :
Pour la syntaxe 1 :

- un nombre d'octets si le type de données du paramètre 3 est binaire ou texte
- un nombre de caractères si le type de données du paramètre 3 est Unicode.

Pour la syntaxe 2 :

- un nombre d'octets si le mode d'ouverture est binaire ou texte
- un nombre de caractères Unicode si le mode d'ouverture est en Unicode.

Exemple :

Cet exemple télécharge un fichier *SAVF vers un répertoire local :
(depuis MY_LIB/MYSAVF vers c:\temp\mysavf.savf)

```
nHdl, cnt est entier
buf est Buffer
nFicLocal est entier

nHdl = ASfOuvre("/QSYS.LIB/MY_LIB.LIB/MYSAVF.FILE", ASfoBinaire)
IF nHdl = -1 ALORS
 Erreur(ErreurInfo())
 RETOUR
END
nFicLocal = fOuvre("c:\temp\mysavf.savf",foEcriture+foCréation)
IF nFicLocal = -1 ALORS
 Erreur(ErreurInfo())
 RETOUR
END

cnt = ASfLit(nHdl, 65000, buf)
TANTQUE cnt<>0
 fEcrit(nFicLocal, buf, cnt)
 cnt = ASfLit(nHdl, 65000, buf)
FIN
fFerme(nFicLocal)
ASfFerme(nHdl)
```

Voir exemple complet.

ASfLitLigne

Lecture d'une ligne dans un fichier IFS.

Syntaxe

```
Résultat = ASfLitLigne(id_ouverture)
```

En anglais : ASfReadLine

Paramètres

Résultat : chaîne de caractères, binaire ou chaîne Unicode

Ligne lue.

Le type de données dépend du mode d'ouverture.

id_ouverture : Entier

Entier fourni par la fonction ASfCree ou ASfOuvre.

Voir exemple complet.

ASfOuvre

Cette fonction permet l'ouverture d'un fichier de l'IFS.

Syntaxe

```
Résultat = ASfOuvre(NomFichier, [,mode] [,Connexion])
```

En anglais : ASfOpen

Paramètres

Résultat : entier

Correspond au numéro d'ouverture du fichier. Cet entier doit être ensuite fourni aux fonctions ASfFerme, ASfLit, ASfEcrit, etc.

Le résultat est -1 en cas d'erreur.

Mode : Entier optionnel

Mode de création du fichier.
La valeur par défaut est : *ASfoLecture*

Connexion : nom ou variable Connexion optionnelle

Nom de la connexion associée.

Exemple

```
nHdl = ASfOuvre("/textes/texte_a_lire.txt",ASfoUnicode)
IF nHdl = -1 ALORS
 Erreur(ErreurInfo())
 RETOUR
END
```

Voir exemple complet.

ASfSauveTexte - ASfSauveBinaire

Ces fonctions permettent de créer un fichier texte ou binaire à partir d'une chaîne ou d'un buffer.

Syntaxe

```
Résultat = ASfSauveTexte(NomFichier, Donnees [,mode] [,Connexion])
Résultat = ASfSauveBinaire(NomFichier, Donnees [,mode] [,Connexion])
```

En anglais : ASfSaveText, ASfSaveBinary

Paramètres

Résultat : booléen

Retour = faux si une erreur est survenue

NomFichier : Chaîne de caractères

Chaîne de caractères contenant le nom du fichier à créer

Donnees : Chaîne de caratères Ansi ou Unicode, Buffer

Chaîne texte ou binaire, buffer contenant le fichier complet à écrire

Mode : Entier optionnel

Mode de création du fichier.
La valeur par défaut est :
ASfoCréation+*ASfoTronque*+*ASfoEcriture*+*ASfoDroitsU_RW*+*ASfoDroitsG_RW*

Connexion : nom ou variable Connexion optionnelle

Nom de la connexion associée à la sauvegarde.

Exemple

```
buf est un Buffer

SI PAS ASfSauveTexte("/tmp/data1.txt", buf, MaConnexion) ALORS
 Info(ErreurInfo())
FIN
```

Mode pour ASfOuvre, ASfCree, ASfSauveTexte/Binaire

Ce mode permet de déterminer la manière d'ouvrir ou créer un fichier IFS pour les fonctions ASfOuvre, ASfCree, ASfSauveTexte/Binaire.

Ce mode est une addition des constantes suivantes:

- type d'ouverture
  - *ASfoAjout* ajout en fin de fichier
  - *ASfoTronque* tronque le fichier
  - *ASfoCréation* créé le fichier, qui ne doit pas exister
  - *ASfoCréationSiInexistant* crée le fichier, s'il n'existe pas
- utilisation
  - *ASfoEcriture* écriture
  - *ASfoLecture* lecture
  - *ASfoLectureEcriture* lecture+écriture
  - *ASfoBinaire* pas de conversions
  - *ASfoUnicode* données texte stockées en Unicode
- verrouillage
  - *ASfoBloqueLecture* blocage en lecture

ASfoBloqueEcriture blocage en écriture

- droits. Il s'agit de droits type Unix, soit la lecture (R), écriture(W), exécution(X) pour l'utilisateur(U), le groupe(G) et les autres(O). Cela reprend les symboliques de la commande Unix chmod.

  Lors de la création d'un fichier, l'utilisateur est toujours l'utilisateur en cours.

  Côté AS/400, la commande CHGAUT permet d'ajuster les droits des objets ifs. CHGOWN permet de modifier l'utilisateur.

  ASfoDroitsU_R le propriétaire du fichier a le droit de lecture

  ASfoDroitsU_W le propriétaire du fichier a le droit d'écriture

  ASfoDroitsU_X le propriétaire du fichier a le droit d'exécution

  ASfoDroitsU_RW = *ASfoDroitsU_R+ASfoDroitsU_W*

  ASfoDroitsU_RX = *ASfoDroitsU_R+ASfoDroitsU_X*

  ASfoDroitsU_RWX = *ASfoDroitsU_RW+ASfoDroitsU_X*

  ASfoDroitsG_R le groupe du fichier a le droit de lecture

  ASfoDroitsG_W le groupe du fichier a le droit d'écriture

  ASfoDroitsG_X le groupe du fichier a le droit d'exécution

  ASfoDroitsG_RW = *ASfoDroitsG_R+ASfoDroitsG_W*

  ASfoDroitsG_RX = *ASfoDroitsG_R+ASfoDroitsG_X*

  ASfoDroitsG_RWX = *ASfoDroitsG_RW+ASfoDroitsG_X*

  ASfoDroitsO_R les autres ont le droit de lecture

  ASfoDroitsO_W le les autres ont le droit d'écriture

  ASfoDroitsO_X le les autres ont le droit d'exécution

  ASfoDroitsO_RW = *ASfoDroitsO_R+ASfoDroitsO_W*

  ASfoDroitsO_RX = *ASfoDroitsO_R+ASfoDroitsO_X*

  ASfoDroitsO_RWX = *ASfoDroitsO_RW+ASfoDroitsO_X*

  ASfoDroitsA_R = ASfoDroitsU_R+ASfoDroitsG_R+ASfoDroitsO_R

  ASfoDroitsA_W = ASfoDroitsU_W+ASfoDroitsG_W+ASfoDroitsO_W

  ASfoDroitsA_X = ASfoDroitsU_X+ASfoDroitsG_X+ASfoDroitsO_X

  ASfoDroitsA_RWX = ASfoDroitsA_R+ASfoDroitsA_W+ASfoDroitsA_X

Exemple

Dans cet exemple, on utilise les fonctions de l'IFS pour créer un fichier sur l'IFS, et écrire dedans: ASfCrée, ASfEcritLigne **et** ASfFerme.

Ensuite, on relit le fichier: ASfOuvre, ASfLitLigne et ASfFerme.

```
nIdOuverture est un entier

nBCarEcritsLigne est un entier
Tmp est une chaîne UNICODE

HOuvreConnexion(MaConnexionpower8)

nIdOuverture =
ASfCrée("/tmp/test_IFS4.txt",ASfoCréation+ASfoTronque+ASfoEcriture+ASfoDroitsU
_RW+ASfoDroitsG_RW+ASfoUnicode)

IF nIdOuverture = -1 ALORS
       Info(ErreurInfo())
       RETOUR
```

```
FIN

Tmp = "Ecriture dans un fichier sur l'IFS"
nBCarEcritsLigne = ASfEcritLigne(nIdOuverture, Tmp)
Info("Nombre de caractères écrits = " + nBCarEcritsLigne)

SI PAS ASfFerme(nIdOuverture) ALORS
	Info(ErreurInfo())
FIN

///// Relecture fichier IFS

nIdOuverture = ASfOuvre("/tmp/test_IFS4.txt",ASfoUnicode)

IF nIdOuverture = -1 ALORS
	Info(ErreurInfo())
	RETOUR
FIN

sChaineLue est une chaîne UNICODE
sChaineLue = ASfLitLigne(nIdOuverture)
Info(sChaineLue)

SI PAS ASfFerme(nIdOuverture) ALORS
	Info(ErreurInfo())
FIN
```

Fonctions Gestion Profil

ASUserList

Récupère la liste des utilisateurs visibles par l'utilisateur de la connexion selon ses droits.

Syntaxe

```
Result = ASUserList(ASUsrPrfCollection,Connexion)
```

En anglais : `ASUserList`

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

AsUsrPrfCollection

Une variable de type ASUsrPrfCollection qui contiendra le résultat de la commande.

C'est une collection d'objets ASUsrPrf.

Connexion

Chaîne – nom qualifié de la connexion

Exemple

```
profilListe est une ASUsrPrfCollection
monProfil est un ASUsrPrf
nb est un entier
```

```
HOuvreConnexion(AS400)
ASUserList(profilListe, AS400)

nb = profilListe.CollectionASUsrPrf..Occurrence
POUR indice= 1 A nb
   monProfil = profilListe.CollectionASUsrPrf[indice]
   TableauAjoute(gtabObjet, monProfil)
FIN
TableAffiche(TABLE_GtabObjet)
```

ASUserListDetailed

Attention ! Cette commande n'est disponible qu'avec l'OS V7R3 côté AS400.

Récupère la liste des utilisateurs visibles par l'utilisateur de la connexion selon ses droits.

Toutes les propriétés du profil de l'utilisateur sont disponibles avec l'objet de type **ASUsrPrfDetailedCollection** en retour de la fonction.

Syntaxe

```
Result = ASUserListDetailed(ASUsrPrfDetailedCollection,Connexion, [UserName],
[StatutUser], [UserGroup], [UserClass],[SpecialAuthority])
```

En anglais : `ASUserListDetailed`

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

AsUsrPrfDetailedCollection

Une variable de type ASUsrPrfDetailedCollection qui contiendra le résultat de la commande.
Il s'agit d'une collection d'objets de type ASUsrPrfDetailed.

Connexion

Chaîne – nom qualifié de la connexion

UserName

Chaîne – Filtre sur le nom des profils à récupérer. Par exemple : * pour tous les utilisateurs.

[optionnel]. Si aucun nom n'est spécifié, les informations de tous les utilisateurs seront récupérées (valeur * ou chaine vide utilisée).

StatutUser

Parmi les constantes suivantes :

ASUtilisateursTousStatuts (en anglais : *ASUsersAllStatus*)
ASUtilisateursDésactivés (en anglais : *ASDisabledUsers*)
ASUtilisateursActivés (en anglais : *ASEnabledUsers*)

[optionnel]. Si aucune valeur n'est spécifiée, la constante *ASUtilisateursTousStatuts* sera utilisée.

UserGroup

Chaîne – Nom du groupe dans lequel on veut récupérer les profils correspondant aux critères de recherche.

[optionnel]. Si aucun groupe n'est spécifié, la recherche va s'opérer sur tous les groupes (valeur * ou chaine vide utilisée).

UserClass

Chaîne – Nom de la user class dans lequel on veut récupérer les profils correspondant aux critères de recherche.

[optionnel]. Si aucune user class n'est spécifiée, la recherche va s'opérer sur toutes les classes (valeur * ou chaine vide utilisée).

SpecialAuthority

Optionnel. Parmi les constantes suivantes :

ASALLOBJ : correspond à *ALLOBJ

ASSECADM : correspond à *SECADM

ASJOBCTL : correspond à *JOBCTL

ASSPLCTL : correspond à *SPLCTL

ASSAVSYS : correspond à *SAVSYS

ASSERVICE : correspond à *SERVICE

ASAUDIT : correspond à *AUDIT

ASIOSYSCFG : correspond à *IOSYSCFG

Un mixage des constantes est possible :
ASALLOBJ+*ASSECADM*+*ASJOBCTL*+*ASSPLCTL*+*ASSAVSYS*+*ASSERVICE*+*ASAUDIT*+*ASIOSYSCFG*

Remarques

1) Les 3 syntaxes suivantes sont équivalentes : on exécute l'opération sur tous les groupes.

ASUserListDetailed(test2,PrinciConnexion,sFiltreNom,ASUtilisateursActivés,"*")
ASUserListDetailed(test2,PrinciConnexion,sFiltreNom,ASUtilisateursActivés,"")
ASUserListDetailed(test2,PrinciConnexion,sFiltreNom,ASUtilisateursActivés)

2) Les 3 syntaxes suivantes sont équivalentes : on exécute l'opération sur toutes les classes pour le groupe « MYGROUP ».

ASUserListDetailed(test2,Connex,sFiltreNom,ASUtilisateursActivés,"MYGROUP", "*")
ASUserListDetailed(test2, Connex,sFiltreNom,ASUtilisateursActivés,"MYGROUP", "")
ASUserListDetailed(test2, Connex,sFiltreNom,ASUtilisateursActivés ,"MYGROUP")

Exemple

```
MaConnexion est une Connexion
// Description de la connexion
MaConnexion..Utilisateur = "trsecofr"
MaConnexion..MotDePasse = "trsecofr"
MaConnexion..Serveur = "power8"

MaConnexion..Provider = hAccèsNatifAS400

MaConnexion..InfosEtendues= "<EASYCOM>"+CRLF+"JOBNAME=CED"+CRLF+"</EASYCOM>"

test2 est une ASUsrPrfDetailedCollection


//Récupérer tous les utilisateurs
SI PAS ASUserListDetailed(test2,MaConnexion) ALORS
       Info(ErreurInfo(errComplet))
       RETOUR
SINON
       DSPUSR();//procédure pour afficher les infos. Voir exemple fourni
```

```
	TableauSupprimeTout(test2)
FIN

//Récupérer tous les utilisateurs
SI PAS ASUserListDetailed(test2,MaConnexion,"*",ASUtilisateursTousStatuts)
ALORS
	Info(ErreurInfo(errComplet))
	RETOUR
SINON
	DSPUSR(); //procédure pour afficher les infos. Voir exemple fourni
	TableauSupprimeTout(test2)
FIN

//Récupérer l'utilisateur dont le nom est QPGMR
HOuvreConnexion(MaConnexion);
SI PAS ASUserListDetailed(test2,MaConnexion,"QPGMR") ALORS
	Info(ErreurInfo(errComplet))
	RETOUR
SINON
	DSPUSR();//procédure pour afficher les infos. Voir exemple fourni
	TableauSupprimeTout(test2)
FIN


//Récupérer tous les utilisateurs désactivés
SI PAS ASUserListDetailed(test2,MaConnexion,"*",ASUtilisateursDésactivés)
ALORS
	Info(ErreurInfo(errComplet))
	RETOUR
SINON
	DSPUSR();//procédure pour afficher les infos. Voir exemple fourni
	TableauSupprimeTout(test2)
FIN

//Récupérer tous les utilisateurs activés
SI PAS ASUserListDetailed(test2,MaConnexion,"*",ASUtilisateursActivés) ALORS

	Info(ErreurInfo(errComplet))
	RETOUR
SINON
	DSPUSR();//procédure pour afficher les infos. Voir exemple fourni
	TableauSupprimeTout(test2)
FIN


//Récupérer tous les utilisateurs dont le nom commence par Q*
SI PAS ASUserListDetailed(test2,MaConnexion,"Q*",ASUtilisateursTousStatuts)
ALORS
	Info(ErreurInfo(errComplet))
	RETOUR
SINON
	DSPUSR();//procédure pour afficher les infos. Voir exemple fourni
	TableauSupprimeTout(test2)
FIN


//Récupérer tous les utilisateurs dont le groupe est : NOGROUP
SI PAS
ASUserListDetailed(test2,MaConnexion,"",ASUtilisateursTousStatuts,"NOGROUP")
ALORS
	Info(ErreurInfo(errComplet))
```

```
        RETOUR
SINON
        DSPUSR();//procédure pour afficher les infos. Voir exemple fourni
        TableauSupprimeTout(test2)
FIN

//Récupérer tous les utilisateurs dont la classe (User Class) est *SECOFR
SI PAS ASUserListDetailed(test2,MaConnexion,"",ASUtilisateursTousStatuts,"",
"*SECOFR") ALORS
        Info(ErreurInfo(errComplet))
        RETOUR
SINON
        DSPUSR();
        TableauSupprimeTout(test2)
FIN

//Récupérer tous les utilisateurs possédant les droits spéciaux « Special
Authorities » spécifiés
SI PAS ASUserListDetailed(test2,MaConnexion,"",ASUtilisateursTousStatuts,"",
"", ASJOBCTL+ASIOSYSCFG) ALORS
        Info(ErreurInfo(errComplet))
        RETOUR
SINON
        DSPUSR();
        TableauSupprimeTout(test2)
FIN
```

ASSignedOnUsersInfoList

Récupère la liste des utilisateurs connectés ou précédemment connectés avec des tâches d'impression en attente.

Syntaxe

```
Result = ASSignedOnUsersInfoList(ASSignedOnUsersInfoCollection, Connexion,
            [NomUtilisateur], [NomStation], [InclureJobsSuspendusDéconnectés],
            [InclureDéconnectésAvecImpressionEnAttente])
```

En anglais : `ASSignedOnUsersInfoList`

Paramètres

Result

Booléen - Vrai si l'opération a été réalisée. Faux en cas de problème. La fonction ErreurInfo permet d'identifier l'erreur.

AsSignedOnUsersInfoCollection

Une variable de type AsSignedOnUsersInfoCollection qui contiendra le résultat de la commande.

C'est une collection d'objets AsSignedOnUsersInfo.

Connexion

Connexion - nom de la connexion

NomUtilisateur

Chaîne de caractères (avec guillemets) – Nom du ou des profill(s) utilisateur(s) recherché(s).

Si vous ne voulez pas filtrer par nom : laissez vide ou mettez "*ALL".

Si vous voulez faire une recherche approximative (commençant par), finissez par * (ex : "QPG*").

NomStation

Chaîne de caractères (avec guillemets) – Nom de la ou des station(s) recherchée(s).

Si vous ne voulez pas filtrer par station : laissez vide ou mettez "*ALL".

Si vous voulez faire une recherche approximative (commençant par), finissez par * (ex : "QPADEV*").

InclureJobsSuspendusDéconnectés

Booléen – Vrai si doivent être inclus les tâches déconnectées et les tâches de groupe suspendues.

Faux sinon.

InclureDéconnectésAvecImpressionEnAttente

Booléen – Vrai si doivent être inclus les utilisateurs autorisés avec une sortie en attente d'impression.

Faux sinon.

Exemple

```
assoic  est un ASSignedOnUsersInfoCollection

ASSignedOnUsersInfoList(assoic,PrinciConnexion,SAI_user,SAI_station,INT_deco,INT_Prin
t)
SAI_Infos = ""
ind est un entier
ind = 1

POUR TOUT assoi DE assoic
       SAI_Infos = SAI_Infos + RC + ind + ")" + RC +
       "Username : " + assoi.userName + RC +
       "User Description : " + assoi.userDesc + RC +
       "Activity : " + assoi.activity + RC +
       "Activity Name : "  + assoi.activityName + RC +
       "Station Name : " + assoi.stationName + RC +
       "Station Desc : " + assoi.displayStationDesc + RC +
       "Disconnected Job Allowed : " + assoi.disconnectJobAllowed + RC +
       "Job Number : " + assoi.jobNumber + RC

       ind++
FIN
```

ASCreerProfil

Crée un nouveau profil AS400. Il faut que le profil utilisé pour ouvrir la connexion, ait les droits nécessaires pour la création d'un profil.

En effet, un profil de type *SECADM (comme QSECOFR par exemple) est obligatoire pour cette fonction.

Syntaxe

```
Result = ASCreerProfil(NomProfil, Mot De Passe, Description, [Profil
Source],Connexion)
```

En anglais : `ASCreateProfil`

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

NomProfil

Chaine de charactères – Le nom du profil AS400 à créer.

Mot de passe

Chaine de charactères – Mot de passe associé au profil AS400 à créer.

Description

Chaine de charactères – Description associé au profil AS400 à créer.

Profil Source

Chaine de charactères – Le nom du profil AS400 à utiliser comme modèle, ce paramètre est optionnel.

Connexion

Chaîne – nom qualifié de la connexion.

Exemple

```
MaConnexion est une Connexion
// Description de la connexion
MaConnexion..Utilisateur = "TSECOFR"
MaConnexion..MotDePasse = "tsecofr"
MaConnexion..Serveur = "power8"

MaConnexion..Provider = hAccèsNatifAS400
HOuvreConnexion(MaConnexion)

SI PAS ASCreerProfil("CRTEST", "crtest", "Test", "QPGMR",MaConnexion) ALORS
       Info(HErreur())
SINON
       Info("Profil créé")
FIN
```

ASModifieProfil

Modifie un profil AS400. Il faut que le profil utilisé pour ouvrir la connexion, ait les droits nécessaires pour la modification d'un profil.

En effet, un profil de type *SECADM (comme QSECOFR par exemple) est obligatoire pour cette fonction.

Syntaxe

```
Result = ASModifieProfil(NomProfil, Champs, Valeur,Connexion)
```

En anglais : `ASModifyProfil`

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

NomProfil

Chaine de charactères – Le nom du profil AS400 à modifier.

Champs

Chaine de charactères – Champs du profil à modifier, les valeurs acceptées sont les suivantes :
PASSWORD,PWDEXP,STATUS,USRCLS,ASTLVL,CURLIB,INLPGM,INLMNU,LMTCPB,TEXT,
SPCAUT,SPCENV,DSPSGNINF,PWDEXPITV,LCLPWDMGT,LMTDEVSSN,KBDBUF,MAXSTG,
PTYLMT,JOBD,GRPPRF,OWNER,GRPAUT,GRPAUTTYP,SUPGRPPRF,ACGCDE,DOCPWD,
MSGQ,DLVRY,SEV,PRTDEV,OUTQ,ATNPGM,SRTSEQ,LANGID,CNTRYID,CCSID,CHRIDCTL,
SETJOBATR,LOCALE,USROPT,UID,GID,HOMEDIR,EIMASSOC.

Valeur

Chaine de charactères – Nouvelle valeur pour le champs.

Connexion

Chaîne – nom qualifié de la connexion.

Exemple

```
MaConnexion est une Connexion
// Description de la connexion
MaConnexion..Utilisateur = "TSECOFR"
MaConnexion..MotDePasse = "tsecofr"
MaConnexion..Serveur = "power8"

MaConnexion..Provider = hAccèsNatifAS400
HOuvreConnexion(MaConnexion)

SI PAS ASModifieProfil("CRTEST", "PASSWORD", "aura",MaConnexion) ALORS
	Info(HErreur())
SINON
	Info("Profil modifié")
FIN
```

ASEffaceProfil

Efface un profil AS400. Il faut que le profil utilisé pour ouvrir la connexion, ait les droits nécessaires pour la suppression d'un profil.

En effet, un profil de type *SECADM (comme QSECOFR par exemple) est obligatoire pour cette fonction.

Syntaxe

```
Result = ASEffaceProfil(NomProfil,Connexion)
```

En anglais : `ASDeleteProfil`

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

NomProfil

Chaine de charactères – Le nom du profil AS400 à effacer.

Connexion

Chaîne – nom qualifié de la connexion.

Exemple

```
MaConnexion est une Connexion
// Description de la connexion
MaConnexion..Utilisateur = "TSECOFR"
MaConnexion..MotDePasse = "tsecofr"
MaConnexion..Serveur = "power8"

MaConnexion..Provider = hAccèsNatifAS400
HOuvreConnexion(MaConnexion)

SI PAS ASEffaceProfil("CRTEST",MaConnexion) ALORS
	Info(HErreur())
SINON
	Info("Profil effacé")
FIN
```

ASVerifieMdP

Vérifie si le mot de passe passé en paramètre est correct.

Syntaxe

```
Result = ASVerifieMdP(NomProfil,MotDePasse,[CCSID],Connexion)
```

En anglais : ASCheckPassword

Paramètres

Result

Booléen - Vrai si le mot de passe est correct, Faux sinon.

NomProfil

Chaine de charactères – Le nom du profil AS400.

MotDePasse

Chaîne – Mot de passe à vérifier.

CCSID

Entier – paramètre optionnel, numéro du CCSID du mot de passe, si il est différent de celui du système.

Connexion

Chaîne – nom qualifié de la connexion.

Exemple

```
MaConnexion est une Connexion
// Description de la connexion
MaConnexion..Utilisateur = "TSECOFR"
MaConnexion..MotDePasse = "tsecofr"
MaConnexion..Serveur = "power8"

MaConnexion..Provider = hAccèsNatifAS400
HOuvreConnexion(MaConnexion)

SI PAS ASVerifieMdP("CRTEST", "aura", MaConnexion) ALORS
       Info("Mot de passe incorrect")
SINON
       Info("Mot de passe correct")
FIN
```

ASVerifieDroits

Vérifie les autorisations d'un profil AS400 sur un objet.

Syntaxe

```
Result = ASVerifieDroits(NomProfil,Objet,Librairie,Typeobjet,Droit,Connexion)
```

En anglais : ASCheckAuthority

Paramètres

Result

Booléen - Vrai si le profil possède le droit demandé sur l'objet, Faux sinon.

NomProfil

Chaine de charactères – Le nom du profil AS400.

Objet

Chaîne – Nom de l'objet sur lequel on teste les droits.

Librairie

Chaîne – Librairie contenant l'objet.

TypeObjet

Chaîne – Type d'objet.

Droit

Chaîne – Droit à verifier, seules les valeurs suivantes sont autorisées :

*EXCLUDE,*ALL,*CHANGE,*USE,*AUTLMGT,OBJALTER,*OBJOPR,*OBJMGT,*OBJEXIST,*OBJREF,*READ,*ADD,*UPD,*DLT,*EXECUTE.

Connexion

Chaîne – nom qualifié de la connexion.

Exemple

```
MaConnexion est une Connexion
// Description de la connexion
MaConnexion..Utilisateur = "TSECOFR"
MaConnexion..MotDePasse = "tsecofr"
MaConnexion..Serveur = "power8"

MaConnexion..Provider = hAccèsNatifAS400
HOuvreConnexion(MaConnexion)

SI PAS ASVerifieDroits("CRTEST", "S_EMPLOYEE", "EASYCOMXMP", "*FILE", "*ALL",
MaConnexion) ALORS
       Info("Le profil n'a pas les droits demandés sur l'objet")
SINON
       Info("Droits accordés")
FIN
```

ASChangeMdp

Change le mot de passe de l'utilisateur courant. Le mot de passe spécifié peut contenir jusqu'à 128 caractères.

Syntaxe

```
Result = ASChangeMdp(AncienMotDePasse, NouveauMotDePasse, Connexion)
```

En anglais : `ASChangePasswd`

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

AncienMotDePasse

Ancien mot de passe pour l'utilisateur courant.

NouveauMotDePasse

Nouveau mot de passe pour l'utilisateur courant.

Connexion

Connexion - nom de la connexion

ASUtilisateur

Cette commande permet de changer d'utilisateur effectif ou d'adopter les droits d'un autre utilisateur.

Elle présente une solution de sécurité facile et souple en donnant au programme les droits d'un profil que l'utilisateur, qui utilise néanmoins son profil pour établir la connexion, ne connaît pas

Une session doit avoir été ouverte au préalable par la fonction HOuvreConnexion.

Dans la liste des jobs c'est toujours l'utilisateur initial qui apparaît mais l'utilisateur courant (CURUSER) est attribué et ses droits sont ajoutés au job. Cette fonction n'est pas conçue pour des changements multiples et répétés.

Il est également possible de ré-attribuer un job à un autre profil côté serveur, par le programme EACTPC003, voir Sécuriser l'accès à EASYCOM.

Syntaxe

```
Result = ASUtilisateur(Nouveau_profil, Nouveau_mot_de_passe [, Connexion])
```

En anglais : ASUser

```
ASUtilisateur("QPGMR","qpgmr",MaConnexion)
```

Paramètres

Result

Booléen - Vrai si le changement d'utilisateur a réussi, Faux en cas d'erreur

Nouveau_profil

Chaîne - nom du nouveau profil

Nouveau_mot_de _passe

Chaîne - mot de passe du nouveau profil

Connexion [optionnel]

Connexion - nom de la connexion

Exemple

```
HOuvreConnexion(MaConnexion, User, Pwd, " 194.206.160.105", hAccèsNatifAS400,
hOLectureEcriture, "")
...
ASUtilisateur(SuperID, SuperPWD)
```

User et Pwd sont saisis par l'utilisateur de l'application. C'est un profil valide qui permet d'ouvrir une session mais avec un minimum de droits. SuperID et SuperPWD ne sont connus que par le développeur. L'administrateur système peut ainsi ne donner aucun droit sur les fichiers aux utilisateurs. Chaque utilisateur devra s'identifier pour se connecter. Une fois connecté, le programme prendra les droits d'un super utilisateur pour accéder aux fichiers.

Objet de type AsUsrPrfDetailedCollection

Ce type d'objet est utilisé en retour de la fonction `ASUserListDetailed`. Il s'agit d'une collection d'objets **AsUsrPrfDetailed**.

Propriétés de l'objet AsUsrPrfDetailed

Nom propriété en Français	Nom propriété en Anglais	Type
NomUtilisateur	UserName	chaîne
Statut	Status	chaîne
DateDernierLoginValide	LastSuccessfulSignOn	DateHeure
NombreTentativesLoginInfructueuses	NumberFailedLoginAttempts	entier sur 4 octets
DateDernierChangementDeMotDePasse	LastPasswordChangeDate	DateHeure
IndicateurMDPestNONE	NONEPasswordIndicator	booléen
IntervalValiditéMDP	PasswordIntervalValidity	chaîne
DateExpirationMDP	PasswordExpirationDate	DateHeure
NbJoursAvantExpirationMDP	NbDayBeforePasswordExpires	entier sur 4 octets
IndicateurExpirationMDP	PasswordExpirationIndicator	booléen
ClasseUtilisateur	UserClass	chaîne
aAuthoritéSpéciale	hasSpecialAuthority	booléen
estALLOBJ	isALLOBJ	booléen
estSECADM	isSECADM	booléen
estJOBCTL	isJOBCTL	booléen
estSPLCTL	isSPLCTL	booléen
estSAVSYS	isSAVSYS	booléen
estAudit	isAUDIT	booléen
estSERVICE	isSERVICE	booléen
estIOSYSCFG	ISIOSYSCFG	booléen
Groupe	Group Profile Name	chaîne
Propriétaire	Owner Name	chaîne
AutoritéDuGroupe	GroupAuthority	chaîne
NiveauAssistance	AssistanceLevel	chaîne
NomCURLIB	CurrentLIBName	chaîne
nomMenuInitial	initialMenuName	chaîne
nomLibMenuInitial	initialMenuLibName	chaîne
nomProgrammeInitial	initialProgramName	chaîne
nomLibProgrammeInitial	initialProgramLibName	chaîne
limitationCapacité	capabilitiesLimitation	chaîne
description	description	chaîne
afficheSignOnInfo	displaySignOnInfo	chaîne
sessionLimitéeAUnPoste	limitDeviceSessions	chaîne
mémoireTamponFrappe	keyboardBuffering	chaîne
maxCapacitéMémoirePermise	maxAllowedStorage	entier sur 4 octets
mémoireUtilisée	storageUsed	entier sur 4 octets

niveauPrioritéMax	highestSchedulingPriority	entier sur 4 octets
nomDescriptionJob	jobDescriptionName	chaîne
nomLibDescriptionJob	jobDescriptionLibName	chaîne
codeComptabilité	accountingCode	chaîne
nomFileDAttenteMessages	messageQueueName	chaîne
nomLibFileDAttenteMessages	messageQueueLibName	chaîne
méthodeReceptionMessage	messageQueueDeliveryMethod	chaîne
gravitéMessageFileAttente	messageQueueSeverity	entier sur 4 octets
nomFileAttenteSortie	outputQueueName	chaîne
nomLibFileAttenteSortie	outputQueueLibName	chaîne
imprimante	printer	chaîne
environnementSpécial	specialEnvironment	chaîne
nomProgrammeATTN	ATTNProgramName	chaîne
nomLibProgrammeATTN	ATTNProgramLibName	chaîne
idLangage	languageID	chaîne
codePaysOuRégion	countryOrRegionID	chaîne
ccsid	ccsid	chaîne
optionsUtilisateur	userOptions	chaîne
estCLKWD	isCLKWD	booléen
estEXPERT	isEXPERT	booléen
estHLPFULL	isHLPFULL	booléen
estSTSMSG	isSTSMSG	booléen
estNOSTSMSG	isNOSTSMSG	booléen
estROLLKEYS	isROLLKEY	booléen
estPRTMSG	isPRTMSG	booléen
nomSéquenceDeTri	SortSequenceTableName	chaîne
libSéquenceDeTri	SortSequenceTableLibName	chaîne
valeursAuditObjet	ObjectAuditingValue	chaîne
valeursAuditActions	UserActionAuditLevel	chaîne
aCMD	hasCMD	booléen
aCREATE	hasCREATE	booléen
aDELETE	hasDELETE	booléen
aJOBDTA	hasJOBDTA	booléen
aOBJMGT	hasOBJMGT	booléen
aOFCSRV	hasOFCSRV	booléen
aOPTICAL	hasOPTICAL	booléen
aPGMADP	hasPGMADP	booléen
aSAVRST	hasSAVRST	booléen
aSECURITY	hasSECURITY	booléen
aSERVICE	hasSERVICE	booléen
aSPLFDTA	hasSPLFDTA	booléen
aSYSMGT	hasSYSMGT	booléen

typeAuthoritéGroupe	GroupAuthorityType	chaîne
nbGroupesSupplémentaires	nbSupplementalGroups	entier sur 4 octets
idUtilisateur	userId	entier non signé sur 4 octets
idGroupe	groupId	entier non signé sur 4 octets
aAucunAttributTravailEnvLocal	hasNoneLocaleJobAttributes	chaîne
aSYSVAL	hasSYSVAL	booléen
aCCSID	hasCCSID	booléen
aDATFMT	hasDATFMT	booléen
aDATSEP	hasDATSEP	booléen
aSRTSEQ	hasSRTSEQ	booléen
aTIMSEP	hasTIMSEP	booléen
aDECFMT	hasDECFMT	booléen
estUnGroupeNonVide	isNotEmptyGroup	entier sur 4 octets
aAuMoinsUnCertificat	hasAtLeastOneDigitalCertificate	booléen
chrid	chrid	chaîne
nbASPDesc	nbASPStorageDescriptor	entier sur 4 octets
gestionMdpLocale	localPasswordManagement	booléen
bloqueChangeMdp	blockPasswordChange	chaîne
estRequisUE	isUERequired	booléen
intervalExpirationUtilisateur	userExpirationInterval	entier sur 4 octets
DateExpirationUtilisateur	dateUserExpiration	DateHeure
actionExpirationUtilisateur	userExpirationAction	chaîne
maxCapacitéMémoirePermiseLong	maxAllowedStorageLong	entier sur 8 octets
mémoireUtiliséeLong	storageUsedLong	entier non signé sur 8 octets

Objet de type AsUsrPrfCollection

Ce type d'objet est utilisé en retour de la fonction `ASUserList`. Il s'agit d'une collection d'objets **AsUsrPrf**.

Propriétés de l'objet AsUsrPrf

Nom propriété	Type
UserName	chaîne
GroupName	chaîne
Text	chaîne
GroupNumber	entier
isGroup	entier
hasGroupMembers	entier

Objet de type AsSignedOnUsersInfoCollection

Ce type d'objet est utilisé en retour de la fonction `ASSignedOnUsersInfoList`. Il s'agit d'une collection d'objets **AsSignedOnUsersInfo**.

Propriétés de l'objet AsSignedOnUsersInfo

Nom propriété	Description	Type
userName	Nom utilisateur	chaîne
userDesc	Description de l'utilisateur	chaîne
activity	Activité	chaîne
activityName	Nom de l'activité	chaîne
stationName	Nom de la station	chaîne
displayStationDesc	Description de la station	chaîne
disconnectJobAllowed	Indicateur de tâche de déconnexion autorisée. 1 : Le travail peut être déconnecté. 0 : Le travail n'est pas autorisé à être déconnecté.	chaîne
jobNumber	Numéro de JOB	chaîne

Fonctions Gestion Imprimantes

ASPrinterList

Récupère la liste des imprimantes enregistrées sur l'AS400 et leurs propriétés, selon les droits de l'utilisateur qui a ouvert la connexion.

Syntaxe

```
Result = ASPrinterList(ASPrinterCollection, Connexion, [NamePrinter], [NameOutputQueue])
```

En anglais : `ASPrinterList`

Paramètres

Result

Booléen - Vrai si la commande a réussi. Faux en cas d'erreur.

Connexion

Chaîne – nom qualifié de la connexion.

AsPrinterCollection

Une variable de type ASPrinterCollection qui contiendra le résultat de la commande.

C'est une collection d'objets ASprinter.

NamePrinter

Chaîne – nom de l'imprimante. Optionnel.

NameOutputQueue

Chaîne – nom qualifié de l'ouput queue. Optionnel.

Exemple : "QUSRSYS/HPPRT"

Exemple

```
aspc est une ASPrinterCollection ;

HOuvreConnexion(MaConnexionpower8)

//No filter
SI PAS AsPrinterList(aspc, MaConnexionpower8) ALORS
	Info(ErreurInfo());
SINON
	DSPPTR(); //procédure pour afficher les infos. Voir exemple fourni
	TableauSupprimeTout(aspc)
FIN

//Name filter (1)
SI PAS AsPrinterList(aspc, MaConnexionpower8,"TEST") ALORS
	Info(ErreurInfo());
SINON
	DSPPTR(); //procédure pour afficher les infos. Voir exemple fourni
	TableauSupprimeTout(aspc)
FIN


//Name filter (2)
SI PAS AsPrinterList(aspc, MaConnexionpower8,"QFQOUTQ;TEST") ALORS
	Info(ErreurInfo());
SINON
	DSPPTR(); //procédure pour afficher les infos. Voir exemple fourni
	TableauSupprimeTout(aspc)
FIN


//Output Queue filter (1)
SI PAS AsPrinterList(aspc, MaConnexionpower8,"", "QUSRSYS/HPDIJON") ALORS

	Info(ErreurInfo());
SINON
	DSPPTR(); //procédure pour afficher les infos. Voir exemple fourni
	TableauSupprimeTout(aspc)
FIN


//Output Queue filter (2)
SI PAS AsPrinterList(aspc, MaConnexionpower8, "",
"QUSRSYS/HPDIJON;*LIBL/QSYSOPR") ALORS
	Info(ErreurInfo());
SINON
	DSPPTR(); //procédure pour afficher les infos. Voir exemple fourni
	TableauSupprimeTout(aspc)
FIN
```

```
//Both Name & Output Queue filter (1 & 1)
SI PAS AsPrinterList(aspc, MaConnexionpower8,"HPDIJON", "QUSRSYS/HPDIJON")
ALORS
	Info(ErreurInfo());
SINON
	DSPPTR(); //procédure pour afficher les infos. Voir exemple fourni
	TableauSupprimeTout(aspc)
FIN


//Both Name & Output Queue filter (2 & 1)
SI PAS AsPrinterList(aspc, MaConnexionpower8, "HPDIJON;TEST",
"QUSRSYS/HPDIJON") ALORS
	Info(ErreurInfo());
SINON
	DSPPTR(); //procédure pour afficher les infos. Voir exemple fourni
	TableauSupprimeTout(aspc)
FIN


//Both Name & Output Queue filter (2 & 2)
SI PAS AsPrinterList(aspc, MaConnexionpower8, "HPDIJON;TEST",
"QUSRSYS/HPDIJON;*LIBL/QSYSOPR") ALORS
	Info(ErreurInfo());
SINON
	DSPPTR(); //procédure pour afficher les infos. Voir exemple fourni
	TableauSupprimeTout(aspc)
FIN
```

Objet de type AsPrinterCollection

Ce type d'objet est utilisé en retour de la fonction `AsPrinterList`. Il s'agit d'une collection d'objets **AsPrinter**.

Propriétés de l'objet AsPrinter

Nom propriété en Français	Nom propriété en Anglais	Type
DeviceName	DeviceName	chaîne
TextDescription	TextDescription	chaîne
OverallStatusCode	OverallStatusCode	entier 4 octets
OverallStatus	OverallStatus	chaîne
DeviceStatusCode	DeviceStatusCode	entier 4 octets
DeviceStatus	DeviceStatus	chaîne
OutputQueueName	OutputQueueName	chaîne
OutputQueueLibrary	OutputQueueLibrary	chaîne
OutputQueueStatus	OutputQueueStatus	chaîne
WriterName	WriterName	chaîne
WriterStatusCode	WriterStatusCode	entier 4 octets

WriterStatus	WriterStatus	chaîne
WriterStarted	WriterStarted	booléen
FormType	FormType	chaîne
CurrentFileName	CurrentFileName	chaîne
CurrentFileUser	CurrentFileUser	chaîne
CurrentFileUserSpecifiedData	CurrentFileUserSpecifiedData	chaîne
isPublishedInNetworkDirectory	isPublishedInNetworkDirectory	booléen

Remarque : OutputQueueName et OutputQueueLibrary auront une valeur si l'imprimante est en cours d'utilisation.

Fonctions Gestion des Partages

ASServerShareInfoList

Récupère la liste des objets partagés (emplacements, imprimantes, output queue...) sur l'AS400 et leurs propriétés, selon les droits de l'utilisateur qui a ouvert la connexion.

Syntaxe

```
Result = AsServerShareInfoList(ASShareInfoCollection, Connexion, [NameFilter])
```

En anglais : `AsServerShareInfoList`

Paramètres

Result

Booléen - Vrai si la commande a réussi. Faux en cas d'erreur.

Connexion

Chaîne – nom qualifié de la connexion.

ASShareInfoCollection

Une variable de type ASShareInfoCollection qui contiendra le résultat de la commande.

C'est une collection d'objets ASShareInfo.

NameFilter

Chaîne – Nom ou partie du nom de l'objet ou des objets que l'on veut lister. Optionnel.

Exemple

```
asServerShareInfoColl est un ASShareInfoCollection

HOuvreConnexion(MaConnexion1)
SI PAS ASServerShareInfoList(asServerShareInfoColl,MaConnexion1) ALORS
       Info(ErreurInfo(errComplet))
SINON
```

```
          POUR TOUT asShareinfo DE asServerShareInfoColl
               Trace(asShareinfo.shareName)
          FIN
FIN

strFilter est une chaîne = "HPDIJ*" //Filtre sur share name
asServerShareInfoColl.CollectionASShareInfo.SupprimeTout()
SI PAS ASServerShareInfoList(asServerShareInfoColl,MaConnexion1,strFilter) ALORS
       Info(ErreurInfo(errComplet))
SINON
          POUR TOUT asShareinfo DE asServerShareInfoColl
               Trace(asShareinfo.shareName)
          FIN
FIN
HFermeConnexion(MaConnexion1)
```

Objet de type ASShareInfoCollection

Ce type d'objet est utilisé en retour de la fonction ASServerShareInfoList. Il s'agit d'une collection d'objets **ASShareInfo**.

Propriétés de l'objet ASShareInfo

Nom propriété	Type
Share name	chaîne
Device type	entier 4 octets
Permissions	entier 4 octets
Maximum users	entier 4 octets
Current users	entier 4 octets
Spooled file type	entier 4 octets
Output Queue Name	chaîne
Output Queue Library	chaîne
Print driver type	chaîne
Text description	chaîne
Path name	chaîne

Fonctions DataQueue

ASEcritDataQueue

Ecriture dans une dataqueue à clé ou non.

Avec la nouvelle syntaxe, la description PCML de la dataqueue est une propriété de l'objet ASDataQueue.

En anglais : ASDataQueueSend

Nouvelle syntaxe recommandée depuis Windev 27

```
Result = ASEcritDataQueue(ASDataQueue [, Connexion], Donnee1 [, Données2,….]
[,Clef])
```

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

ASDataQueue

Objet de type ASDataQueue contenant la spécification de la data queue (description PCML, nom, bibliothèque…).

La syntaxe PCML est la même que celle utilisée pour ASAppelPgm.

Connexion

Connexion - nom de la connexion. Optionnel.

Donnée1

Variable ou valeurs correspondant aux données à envoyer, de description compatible.

Donnée2….

Variable ou valeurs correspondant aux données à envoyer, de description compatible. Optionnel.

Clef

Chaîne - Valeur de la clé. Optionnel.

Exemples avec la nouvelle syntaxe

1) Dataqueue avec clé

```
sdq_pcml est chaîne =[
	<pcml version="4.0">
	<program name="TESTDQ" >
	<data name="data" type="char" length="200" usage="inputoutput"/>
	<data name="key" type="char" length="10" usage="inputoutput"/>
	</program>
	</pcml>
]


sClé = SAI_Clé1

asdq est une ASDataQueue
asdq.description		= sdq_pcml
asdq.library			= "EASYCOMXMP"
asdq.name			= "DTAQ_KEY"
asdq.timeout			= 0


SI PAS ASEcritDataQueue(asdq, SAI_valeur1, sClé) ALORS
	Info(ErreurInfo(errComplet))
SINON
	Info("Ecriture effectuée")
FIN
```

Voir Exemple.

2) Dataqueue FIFO

```
sdq_pcml est chaîne =[
       <pcml version="4.0">
       <program name="TESTDQ" >
       <data name="data" type="char" length="50" usage="input"/>
       </program>
       </pcml>
]


asdq est une ASDataQueue
asdq.description            = sdq_pcml
asdq.library                = "EASYCOMXMP"
asdq.name                   = "DTAQ_FIFO"
asdq.timeout                = 0

SI PAS ASEcritDataQueue(asdq, PrinciConnexion,SAI_valeur3) ALORS
       Info(ErreurInfo(errComplet))
SINON
       Info("Ecriture effectuée")
FIN
```

Voir Exemple.

Ancienne syntaxe

```
Result = ASEcritDataQueue(DéfinitionDataQueue, NomDataQueue [, Connexion],
Donnee1 [, Données2,….] [,Clef])
```

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

DéfinitionDataQueue

Une chaîne multi-ligne contenant le source PCML correspondant à la description de la donnée et de la clé (voir exemple ci-dessous).

La syntaxe PCML est la même que celle utilisée pour ASAppelPgm.

NomDataQueue

Chaîne – nom qualifié de la dataqueue.

Connexion

Connexion - nom de la connexion.

Donnée1

Variable ou valeur correspondant aux données à envoyer, de description compatible.

Donnée2….

Variable ou valeurs correspondant aux données à envoyer, de description compatible. Optionnel.

Clef

Chaîne - Valeur de la clé. Optionnel.

Exemple avec l'ancienne syntaxe

```
       sdq_pcml est chaîne = [
```

```
<pcml version="4.0">
<program name="ExampleDQK" >
<data name="data" type="char" length="200" usage="input"/>
<data name="key" type="char" length="10" usage="input"/>
</program>
</pcml>
]

SI PAS ASEcritDataQueue(sdq_pcml, "*LIBL/DTAQ_KEY", MaConnexion, sDon,
sCle) ALORS
 Erreur("Echec écriture")
FIN
```

ASLitDataQueue

Lecture d'une dataqueue, sans clé.

Avec la nouvelle syntaxe, la description PCML de la dataqueue est une propriété de l'objet ASDataQueue.

En anglais : `ASDataQueueReceive`

Nouvelle syntaxe recommandée depuis Windev 27

```
Result = ASLitDataQueue (ASDataQueue [, Connexion], Param1 [, Param2,….] )
```

Result

Booléen - Vrai si la commande a réussi. Faux en cas d'erreur.

ASDataQueue

Objet de type ASDataQueue contenant la spécification de la data queue (description PCML, nom, bibliothèque…).

La syntaxe PCML est la même que celle utilisée pour ASAppelPgm.

Connexion

Connexion - nom de la connexion. Optionnel.

Param1

Variable ou valeurs correspondant aux données à lire.

Param2, …

Variable ou valeurs correspondant aux données à lire. Optionnel.

Exemple avec la nouvelle syntaxe

Dataqueue FIFO

```
sdq_pcml est chaîne =[
       <pcml version="4.0">
       <program name="TESTDQ" >
       <data name="data" type="char" length="50" usage="input"/>
       </program>
       </pcml>
]
```

```
data est chaîne sur 50

asdq est une ASDataQueue
asdq.description            = sdq_pcml
asdq.library                = "EASYCOMXMP"
asdq.name                   = "DTAQ_FIFO"
asdq.timeout                = 0
asdq.remove                 = true


SI PAS ASLitDataQueue(asdq, data) ALORS
       Info(ErreurInfo(errComplet))
SINON
       Info(data)
FIN
```

Voir Exemple.

Ancienne syntaxe

```
Result = ASLitDataQueue (DéfinitionDataQueue, NomDataQueue, Timeout, Enlever, [,
Connexion], Param1 [, Param2,….] )
```

Result

Booléen - Vrai si la commande a réussi. Faux en cas d'erreur.

DéfinitionDataQueue

Une chaîne multi-ligne contenant le source PCML correspondant à la description de la donnée.

La syntaxe PCML est la même que celle utilisée pour ASAppelPgm.

NomDataQueue

Chaîne – nom qualifié de la dataqueue

Timeout

Entier - Délai d'attente au cas où la file serait vide à l'appel.

Enlever

Booleen – Supprimer l'entrée au moment de la lecture.

Connexion

Connexion - nom de la connexion.

Param1

Variable ou valeur correspondant aux données à lire.

Parm2 , …

Variables ou valeurs correspondant aux données à lire. Optionnel.

Exemple avec l'ancienne syntaxe

```
sdq_pcml est chaîne =[
<pcml version="4.0">
<program name="TESTDQ" >
<data name="data" type="char" length="50" usage="input"/>
</program>
</pcml>
]
```

```
SI PAS ASLitDataQueue(sdq_pcml, "*LIBL/DTAQ_FIFO", 2, True ,MaConnexion,
data_lue) ALORS
 Info("Data Queue vide!")
FIN
```

ASLitDataQueueCle

Lecture d'une dataqueue à clé.

Avec la nouvelle syntaxe, la description PCML de la dataqueue est une propriété de l'objet ASDataQueue.

En anglais : ASKeyDataQueueReceive

Nouvelle syntaxe recommandée depuis Windev 27

```
bResult = ASLitDataQueueCle(ASDataQueue, Opérateur, [, Connexion], Donnee1 [,
Donnee2….], Clef)
```

bResult

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

ASDataQueue

Objet de type ASDataQueue contenant la spécification de la data queue (description PCML, nom, bibliothèque…).

La syntaxe PCML est la même que celle utilisée pour ASAppelPgm.

Opérateur

Chaîne – Opérateur à utiliser pour la clé. Les valeurs possibles sont :

EQ : égal, NE : différent, LE : <=, LT : <, GE : >=, GT : >.

Connexion

Connexion - nom de la connexion. Optionnel.

Donnee1

Variable ou valeurs correspondant aux données à lire, de description compatible.

Donnee2

Variable ou valeurs correspondant aux données à lire, de description compatible. Optionnel.

Clef

Chaîne - Valeur de la clé

Exemple avec la nouvelle syntaxe

```
sdq_pcml est chaîne =[
       <pcml version="4.0">
       <program name="TESTDQ" >
       <data name="data" type="char" length="200" usage="inputoutput"/>
       <data name="key" type="char" length="10" usage="inputoutput"/>
       </program>
       </pcml>
]



data est chaîne sur 50
```

```
asdq est une ASDataQueue
asdq.description            = sdq_pcml
asdq.library                = "EASYCOMXMP"
asdq.name                         = "DTAQ_KEY"
asdq.timeout                = 0
asdq.remove                 = true


SI PAS ASLitDataQueueCle(asdq, "EQ", data, SAI_Clé1) ALORS
       Info(ErreurInfo(errComplet))
SINON
       Info(data)
FIN
```

Voir Exemple.

Ancienne syntaxe

```
Result = ASLitDataQueueCle(DéfinitionDataQueue, NomDataQueue, Opérateur, Timeout,
Enlever [, Connexion], Donnee1 [, Donnee2…], Clef)
```

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

DéfinitionDataQueue

Une chaîne multi-ligne contenant le source PCML correspondant à la description de la donnée et de la clé (voir exemple ci-dessous)

La syntaxe PCML est la même que celle utilisée pour ASAppelPgm.

NomDataQueue

Chaîne – nom qualifié de la dataqueue.

Opérateur

Chaîne – Opérateur à utiliser pour la clé. Les valeurs possibles sont :

EQ : égal, NE : différent, LE : <=, LT : <, GE : >=, GT : >.

Timeout

Entier - Délai d'attente au cas où la file serait vide à l'appel.

Enlever

Booleen – Supprimer l'entrée au moment de la lecture

Connexion

Connexion - nom de la connexion. Optionnel.

Donnee1

Variable ou valeurs correspondant aux données à lire, de description compatible.

Donnee2

Variable ou valeurs correspondant aux données à lire, de description compatible. Optionnel.

Cle

Chaîne - Valeur de la clé

Exemple avec l'ancienne syntaxe

```
sdq_pcml est chaîne =
<pcml version=="4.0">
<program name="TESTDQ" >
<data name="data" type="char" length=200" usage="input"/>
```

```
<data name="key" type="char" length="10" usage="input"/>
</program>
</pcml>
]

SI PAS ASLitDataQueueCle(sdq_pcml, "*LIBL/DTAQ_KEY", "EQ", 2, True
,MaConnexion, data_lue, key2) ALORS
 Info("Data Queue vide!")
FIN
```

ASDataQueueInfo

Permet de retrouver des informations concernant la data queue spécifiée.

Ces informations sont disponibles dans l'objet de type ASDataQueueInfo, passé en paramètre.

Syntaxe

```
Result = ASDataQueueInfo(ASDataQueueInfo, Connexion, NomDataQueue,
[BibliothèqueDataQueue])
```

En anglais : ASDataQueueInfo

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

ASDataQueueInfo

Objet de type ASDataQueueInfo contenant, en retour de la commande, les infos de la data queue spécifiée.

Connexion

Connexion - nom de la connexion.

NomDataQueue

Chaîne – nom de la dataqueue.

BibliothèqueDataQueue

Chaîne – Bibliothèque contenant la dataqueue. Optionnel si cette bibliothèque fait déjà partie de la liste des bibliothèques du JOB correspondant à la connexion (*LIBL).

Exemple

1) "DTAQ_KEY" fait partie de *LIBL

```
asdqi    est un ASDataQueueInfo

SI PAS ASDataQueueInfo(asdqi,PrinciConnexion,"DTAQ_KEY") ALORS
       Info(ErreurInfo(errComplet))
SINON
       SAI_DQK = "LIBRARY = " + asdqi.dataQueueLib + RC + "Type = " +
asdqi.sequenceDesc + RC + "Description = " + asdqi.dataQueueDescription +
       RC + "Number of messages = " + asdqi.nbMessage + RC + "Key length = " +
asdqi.keyLength
FIN
```

2) Spécification de la bibliothèque contenant la dataqueue

```
asdqi    est un ASDataQueueInfo

SI PAS ASDataQueueInfo(asdqi,PrinciConnexion,SAI_DQname, SAI_DQlib) ALORS
       Info(ErreurInfo(errComplet))
SINON
       SAI_DQK2 = "LIBRARY = " + asdqi.dataQueueLib + RC + "Type = " +
asdqi.sequenceDesc + RC + "Description = " + asdqi.dataQueueDescription +
       RC + "Number of messages = " + asdqi.nbMessage
FIN
```

ASDataQueueClear

Permet de vider le contenu de la data queue spécifiée via l'objet ASDataQueue

Syntaxe

```
Result = ASDataQueueClear(ASDataQueue, Connexion)
```

En anglais : `ASDataQueueClear`

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

ASDataQueue

Objet de type ASDataQueue.

Connexion

Connexion - nom de la connexion.

Exemple

```
asdq est une ASDataQueue
asdq.description            = sdq_pcml
asdq.library                = "EASYCOMXMP"
asdq.name                   = "DTAQ_KEY"
asdq.timeout                = 0

SI PAS ASDataQueueClear(asdq, PrinciConnexion) ALORS
Info(ErreurInfo(errComplet))
FIN
```

Objet de type ASDataQueue

Propriétés de l'objet ASDataQueue

Nom propriété	Description	Type

description	Description dataqueue. PCML.	chaîne
library	Bibliothèque dataqueue	chaîne
name	Nom dataqueue	chaîne
timeout	Timeout. Pour la lecture, délai d'attente au cas où la file serait vide à l'appel.	entier
remove	Effacer la donnée lue.	booléen

Objet de type ASDataQueueInfo

Ce type d'objet est utilisé en retour de la fonction `ASDataQueueInfo`.

Propriétés de l'objet ASDataQueueInfo

Nom propriété	Description	Type
dataQueueDescription	Description dataqueue	chaîne
dataQueueLib	Bibliothèque dataqueue	chaîne
dataQueueName	Nom dataqueue	chaîne
initialNbEntries	Nombre initial d'entrées	entier
isAutoReclaimed	Récupérartion automatique	booléen
isDDMDataQueue	Description de la station	booléen
isEnforcedDQLocks	Appliquer des verrous de file d'attente de données	booléen
isForceAuxStorage	Si la file d'attente de données est forcée ou non vers le stockage auxiliaire lorsque des entrées sont envoyées ou reçues pour la file d'attente de données spécifiée.	booléen
isSenderIdIncluded	Inclure l'identifiant de l'expéditeur.	booléen
keyLength	Longueur de la clé.	entier
lastReclaimDateTime	Date et heure de la dernière récupération.	date
maxNbEntriesSpecified	Nombre maximum d'entrées spécifié.	entier
messageLength	Longueur message.	entier
nbAllocatedEntries	Nombre d'entrées actuellement attribuées.	entier
nbMaxAllowedEntries	Nombre maximum d'entrées autorisées.	entier
nbMessage	Nombre de messages.	entier
sequence	Séquence.	chaîne

sequenceDesc	Description séquence.	chaîne

Exemples

1) Ecriture et lecture – Data Queue avec clé

```
sdq_pcml est chaîne =[
	<pcml version="4.0">
	<program name="TESTDQ" >
	<data name="data" type="char" length="200" usage="inputoutput"/>
	<data name="key" type="char" length="10" usage="inputoutput"/>
	</program>
	</pcml>
]

HOuvreConnexion(MaConnexionpower8)

sClé = SAI_Clé1

asdq est une ASDataQueue
asdq.description            = sdq_pcml
asdq.library                = "EASYCOMXMP"
asdq.name                   = "DTAQ_KEY"
asdq.timeout                = 0
asdq.remove                 = true



SI PAS ASEcritDataQueue(asdq, SAI_valeur1, sClé) ALORS
	Info(ErreurInfo(errComplet))
SINON
	Info("Ecriture effectuée")
FIN

//Read
data est chaîne sur 50

SI PAS ASLitDataQueueCle(asdq, "EQ", data, SAI_Clé1) ALORS
	Info(ErreurInfo(errComplet))
SINON
	Info(data)
FIN
```

2) Ecriture et lecture – Data Queue sans clé

```
sdq_pcml est chaîne =[
	<pcml version="4.0">
	<program name="TESTDQ" >
	<data name="data" type="char" length="50" usage="input"/>
	</program>
	</pcml>
]

HOuvreConnexion(MaConnexionpower8)

asdq est une ASDataQueue
```

```
asdq.description            = sdq_pcml
asdq.library                = "EASYCOMXMP"
asdq.name                   = "DTAQ_FIFO"
asdq.timeout                = 0
asdq.remove                 = true


SI PAS ASEcritDataQueue(asdq, PrinciConnexion,SAI_valeur3) ALORS
       Info(ErreurInfo(errComplet))
SINON
       Info("Ecriture effectuée")
FIN



//Read
data est chaîne sur 50

SI PAS ASLitDataQueue(asdq, data) ALORS
       Info(ErreurInfo(errComplet))
SINON
       Info(data)
FIN
```

Fonctions UserSpace

ASEcritUserSpace

Cette fonction permet d'écrire une donnée structurée ou non dans un userspace.

Syntaxe

```
Result = ASEcritUserSpace(DescUserSpace, NomUserSpace, Offset [, Connexion],
Param1 [, Param2 [, Parm3, ...])
```

En anglais : `ASUserSpaceWrite`

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

DescUserSpace

Une chaîne multi-ligne contenant le source PCML correspondant à la description du userspace.

La syntaxe PCML est la même que celle utilisée pour ASAppelPgm.

NomUserSpace

Chaîne – correspond au nom réel du userSpace

Connexion [optionnel]

Connexion - nom de la connexion

Offset

Entier – correspond à l'indice (départ 1) à partir duquel écrire. La description embarquée commence à cet indice. Cela permet de ne pas avoir à décrire tout le userspace.

Param1, Parm2 , …

Variable ou valeurs correspondant aux données à écrire.

Exemple

```
ch est chaîne

sus_pcml est chaîne = [
<pcml version="4.0">
<program name="TESTUS" >
<data name="data" type="char" length="45" usage="input"/>
</program>
</pcml>
]


ch="test uspc"

SI PAS ASEcritUserSpace(sus_pcml, "MY_LIB/MY_US", 1, ch)
 Erreur(ErreurInfo())
FIN
```

Voir Exemple.

ASLitUserSpace

Cette fonction permet de lire une donnée structurée ou non dans un userspace.

Syntaxe

```
Result = ASLitUserSpace(DescUserSpace, NomUserSpace, Offset [, Connexion], Param1
[, Param2 [, Parm3, ...])
```

En anglais : ASUserSpaceRead

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

DescUserSpace

une chaîne multi-ligne contenant le source PCML correspondant à la description du userspace.

La syntaxe PCML est la même que celle utilisée pour ASAppelPgm.

NomUserSpace

Chaîne – correspond au nom réel du userSpace

Connexion [optionnel]

Connexion - nom de la connexion

Offset

Entier – correspond à l'indice (départ 1) à partir duquel lire. La description embarquée commence à cet indice. Cela permet de ne pas avoir à décrire toute le userspace.

Param1, Parm2 , …

Variable ou valeurs correspondant aux données à lire.

Exemple

```
ch est chaîne

sus_pcml est chaîne =
```

```
<pcml version="4.0">
<program name="US_TEST" >
<data name="data" type="char" length="45" usage="input"/>
</program>
</pcml>
]

SI PAS ASLitUserSpace(sus_pcml, "MY_LIB/MY_US", 1, ch)
 Erreur(ErreurInfo())
FIN
```

Voir Exemple.

ASCreeUserSpace

Cette fonction permet de créer un userspace.

Syntaxe

```
Result = ASCreeUserSpace(Connexion, NomUserSpace, NomLibrary)
```

En anglais : `ASUserSpaceCreate.`

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

Connexion

Connexion - nom de la connexion

NomUserSpace

Chaîne – correspond au nom réel du userSpace

NomLibrary

Chaîne – correspond au nom réel de la bibliothèque qui contiendra le userspace.

Exemple

```
SI PAS ASCreeUserSpace(MaConnexionpower8,"USPC1","CR") ALORS
       Info(ErreurInfo())
FIN
```

Voir Exemple.

ASInfoUserSpace

Cette fonction permet de récupérer les infos du userspace.

Les infos récupérées sont stockées dans un objet de type **ASUserSpaceInfo**.

Syntaxe

```
Result = ASInfoUserSpace(UserSpaceInfo, Connexion, NomUserSpace, NomLibrary)
```

En anglais : `ASInfoUserSpace.`

Paramètres

Result
Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

UserSpaceInfo
Objet de type ASUserSpaceInfo qui contiendra les infos du userspace spécifié.

Connexion
Connexion - nom de la connexion.

NomUserSpace
Chaîne – correspond au nom réel du userSpace.

NomLibrary
Chaîne – correspond au nom réel de la bibliothèque qui contient le userspace.

Exemple

```
infoUSPC est un ASUserSpaceInfo

SI ASInfoUserSpace(infoUSPC,MaConnexionpower8,"USPC1","*LIBL") ALORS

	SI infoUSPC.autoExtend ALORS

		isAutoExtend = Vrai

	FIN

	Info("USPC LIB : " + infoUSPC.uspcLibName,"USPC Space : " +
infoUSPC.spaceSize,"USPC AutoExtend : " + isAutoExtend ,"USPC Initial Char value : "
+ infoUSPC.InitialValue);

SINON
	Info(ErreurInfo())
FIN
```

Voir Exemple.

ASSupprUserSpace

Cette fonction permet de supprimer un userspace.

Syntaxe

```
Result = ASSupprUserSpace(Connexion, NomUserSpace, NomLibrary)
```

En anglais : `ASUserSpaceDelete.`

Paramètres

Result
Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

Connexion
Connexion - nom de la connexion.

NomUserSpace

Chaîne - correspond au nom réel du userSpace.

NomLibrary

Chaîne - correspond au nom réel de la bibliothèque qui contient le userspace.

Exemple

```
ASSupprUserSpace(MaConnexionpower8,"USPC1","CR")
```

Voir Exemple.

Objet de type ASUserSpaceInfo

Ce type d'objet est utilisé en retour de la fonction **ASInfoUserSpace**.

Propriétés

autoExtend	Extensibilité automatique (true/false)
uspcLibName	La bibliothèque dans laquelle se trouve le userpace. Ceci est utile lorsque *LIBL ou *CURLIB est spécifié comme nom de bibliothèque dans le paramètre NomLibrary de la fonction ASInfoUserSpace.
spaceSize	La taille du userspace en bytes.
InitialValue	Valeur initiale avec laquelle les futures extensions du userspace seront définies. Çela peut être un espace, un 0, etc…

Exemple

Création d'un user space, récupération des infos, écriture, lecture et suppression.

```
ch est chaîne

sus_pcml est chaîne = [
	<pcml version=="4.0">
	<program name="TESTUS" >
	<data name="data" type="char" length="45" usage="input"/>
	</program>
	</pcml>
]

HOuvreConnexion(MaConnexionpower8)

//Création
SI PAS ASCreeUserSpace(MaConnexionpower8,"USPC1","CR") ALORS
	Info(ErreurInfo())
FIN

isAutoExtend est un booléen = Faux
infoUSPC est un ASUserSpaceInfo

//Récupération des infos
SI ASInfoUserSpace(infoUSPC,MaConnexionpower8,"USPC1","*LIBL") ALORS

	SI infoUSPC.autoExtend ALORS
```

```
            isAutoExtend = Vrai

        FIN

        Info("USPC LIB : " + infoUSPC.uspcLibName,"USPC Space : " +
infoUSPC.spaceSize,"USPC AutoExtend : " + isAutoExtend ,"USPC Initial Char value : "
+ infoUSPC.InitialValue);

SINON
        Info(ErreurInfo())
FIN

ch="test uspc"

//Ecriture
SI PAS ASEcritUserSpace(sus_pcml, "CR/USPC1", 1, MaConnexionpower8,ch)
        Erreur(ErreurInfo())
FIN

sRes est une chaîne

//Lecture
SI PAS ASLitUserSpace(sus_pcml, "CR/USPC1", 1, MaConnexionpower8, sRes)
        Erreur(ErreurInfo())
SINON
        Info(sRes)
FIN

//Suppression
ASSupprUserSpace(MaConnexionpower8,"USPC1","CR")
```

Fonctions Spool

ASCreeSpool

Crée un fichier spool de type binaire sur l'AS/400 à partir d'un fichier d'impression.

Cette fonction rend possible l'impression d'un état vers un spool AS/400.

Syntaxe

```
Result = ASCreeSpool(NomFichier_prn, Connexion, options [, user_data, user_name,
user_text [, nb_copies, nom_spool [, output_queue, outpout priority, form_type]]] )
```

En anglais : `ASCreateSpool`.

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

NomFichier_prn

Fichier d'impression au format de l'imprimante destination (PCL ou autre). Utiliser `iDestination(iFichier, « nom_fichier »)` pour créer le fichier d'impression à partir d'une impression WinDev classique.

Connexion

Connexion - nom de la connexion

Options

Options de base du spool. Les valeurs possibles sont une combinaison des constantes suivantes :

ASsplHold Créer le spool en état suspendu.

ASsplSave Sauvegarde du spool après impression

User_data (10 caractères)

Donnée utilisateur. Identifiant court de l'impression. Appelée aussi reference utilisateur. Permet de retrouver rapidement le spool par la suite.

User_name

Nom de l'utilisateur devenant propriétaire du spool. Valeurs possibles :

- nom d'utilisateur
- *CURRENT (par défaut) -> utilisateur actif

User_text (100 caractères)

Texte utilisateur. Permet de placer un descriptif personnalisé.

Nb_copies

Nombre de copies à imprimer

Nom_spool (10 caractères)

Nom du fichier spool à créer

Output_Queue

Nom de la file de sortie à utiliser. Valeurs possibles :

- Nom d'une file de données (*LIBL/OUTQ, ou OUTQ)
- *JOB (par défaut)

Output_priority

Priorité de sortie. Valeurs possibles :

1-9

*JOB (par défaut)

Form_type (10 caractères)

Type d'imprimé. Valeurs possibles : *STD ou chaine libre

Exemple

```
// redirige l'impression de ETAT_Etat1 vers un fichier
iDestination(iFichier, fRépertoireTemp()+"etat_impr.prn")
iImprimeEtat(ETAT_Etat1)

// exemple 1
SI PAS ASCreeSpool(fRépertoireTemp()+"etat_impr.prn", MaConnexion1,0,
"QPGMR", "DOC0099", "Facture client XYZ") ALORS
 Erreur(ErreurInfo(errComplet))
FIN

// exemple 2
```

```
SI PAS ASCreeSpool(fRépertoireTemp()+"etat_impr.prn",
MaConnexion1,ASsplHold+ASsplSave, "*CURRENT", "DOC0099", "Facture client
XYZ",2, "MONSPOOL","PRT_EAC", 1, "FRM01") ALORS
 Erreur(ErreurInfo(errComplet))
FIN
```

ASGetSpool

Lit un spool AS/400 vers un fichier PC local PCL ou texte.

Syntaxe

```
Result = ASGetSpool(NomFichier_local, Connexion , Nom_Spool [, job [, spl_number [,
spl_format [, paper_size [, supprLF ]]]]] )
```

En anglais : `ASGetSpool.`

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

NomFichier_Local

Nom du fichier à créer sur le PC.

Connexion

Connexion - nom de la connexion.

Nom_Spool (10 caractères)

Nom du fichier spool à lire. Valeurs possibles : *LAST ou nom.

Job

Identifie le job ayant créé le fichier spool. Les valeurs possibles sont :

* : job en cours

nnnn/USER/NOM_JOB : job numéro nnn utilisateur USER job NOM_JOB

spl_number

numéro du spool. Les valeurs possibles sont :

- de 0 à 999999 : numéro du spool
- `ASsplOnly (*ONLY)`
- `ASsplLast (*LAST)`

Spl_format

Format à utiliser pour le résultat. Les possibles sont :

*TXT si le spool est au fomat texte

*PCL si le spool est au format PCL

Paper_size

Taille du papier. Les valeurs possibles sont :

*NONE *MFRTYPMDL *LETTER *LEGAL *EXECUTIVE *A3 *A4 *A5 *B4 *B5 *CONT80 *CONT132

supprLF (booléen)

Suprimer le dernier saut de page du résultat (par défaut : faux)

Exemple

```
SI PAS ASGetSpool("c:\temp\spl_test_wd.spl", MaConnexion1, "QPJOBLOG",
"108755/MARTIN/PRTXJK", 1, ASsplPCL, "", False)
```

```
 Erreur(ErreurInfo(errComplet))
FIN
```

ASSpoolList

Liste les fichiers spool AS/400 d'une file d'attente de sortie, ou d'un utilisateur.

Syntaxe

```
Result = ASSpoolList(CollectionDeASSpool, Utilisateur,  File d'attente,  Clé de
donnée, Travail, Numéro du travail, Connexion)
```

En anglais : `ASSpoolList.`

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

CollectionDeAsSpool

Variable de type AsSpoolCollection qui va contenir les résultats.

Utilisateur

Nom de l'utilisateur qui a créé le travail.

File d'attente

Nom qualifié de la file d'attente de sortie contenant le fichier Spool.

Clé de donnée

Clé de donnée spécifique à l'utilisateur pour le fichier Spool.

Travail

Nom du travail qui a généré le fichier Spool.

Numéro de travail

Numéro du travail qui a généré le fichier Spool.

Connexion

Connexion - nom de la connexion

Exemple

```
listSpool est un ASSpoolCollection
//recherche toute les fichiers spool AS/400
SI PAS ASSpoolList(listSpool,"","", "","","", MaConnexion1)
 Erreur(ErreurInfo(errComplet))
FIN
```

Objet de type ASSpoolCollection

Ce type d'objet est utilisé en retour de la fonction `ASSpoolList` . Il s'agit d'une collection d'objets **ASSpool.**

Propriétés de l'objet ASSpool

Propriété	Type AS400

Job name	CHAR(10)
User name	CHAR(10)
Job number	CHAR(6)
Spooled file name	CHAR(10)
Spooled file number	BINARY(4)
Spooled file status	BINARY(4)
Date file was opened (created)	CHAR(7)
Time file was opened (created)	CHAR(6)
Spooled file schedule	CHAR(1)
Spooled file system name	CHAR(10)
User-specified data	CHAR(10)
Form type	CHAR(10)
Output queue name	CHAR(10)
Output queue library name	CHAR(10)
Auxiliary storage pool	BINARY(4)
Size of spooled file	BINARY(4)
Spooled file size multiplier	BINARY(4)
Total pages	BINARY(4)
Copies left to produce	BINARY(4)
Priority	CHAR(1)

Fonctions Gestion de Traces

ASActiveTrace

Active ou désactive la trace AS400 utilisée pour regarder l'activité du serveur Easycom.

Syntaxe

```
Result = ASActiveTrace(ActiveTrace, FichierTrace, NiveauTrace, Horodatage, Connexion)
```

En anglais : `ASSetTrace.`

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

ActiveTrace

Booléen - Si vrai on active la trace, si faux on la désactive.

FichierTrace

Nom du fichier de trace à utiliser, il doit respecter le format librairie/fichier.

NiveauTrace

Niveau de détail de la trace, entre 1 et 4, 4 offrant le maximum de détail.

Horodatage

Booléen - Si vrai chaque ligne de la trace sera précédée d'un timestamp, si faux, pas de timestamp.

Connexion

Connexion - Nom de la connexion.

Exemple

```
//active la trace
SI PAS ASActiveTrace(Vrai,"EASYCOM/TRACE",4, Vrai, MaConnexion1)
 Erreur(ErreurInfo())
FIN
```

ASSiTraceActive

Vérifie si la trace AS400 est activée.

Syntaxe

```
Result = ASSiTraceActive(Connexion)
```

En anglais : `ASifTraceActive.`

Paramètres

Result

Booléen - Vrai si la la trace est activée, Faux sinon.

Connexion

Connexion - Nom de la connexion.

Voir Exemple.

ASGetTraceLib

Retourne le nom de la librairie utilisée pour le fichier trace AS400.

Syntaxe

```
Result = ASGetTraceLib(Connexion)
```

En anglais : `ASGetTraceLib.`

Paramètres

Result

Chaine de caractères – Nom de la librairie utilisée pour la trace.

Connexion

Connexion - Nom de la connexion

Voir Exemple.

ASGetTraceFic

Retourne le nom du fichier utilisé pour le fichier trace AS400.

Syntaxe

```
Result = ASGetTraceFic(Connexion)
```

En anglais : `ASGetTraceFic.`

Paramètres

Result

Chaine de caractères – Nom de du fichier utilisé pour la trace.

Connexion

Connexion - Nom de la connexion.

Voir Exemple.

ASGetTraceLvl

Vérifie le niveau de détail utilisé pour le fichier trace AS400.

Syntaxe

```
Result = ASGetTraceLvl(Connexion)
```

En anglais : `ASGetTraceLvl.`

Paramètres

Result

Entier – Niveau de détail utilisé pour la trace.

Connexion

Connexion - Nom de la connexion.

Voir Exemple.

Exemple

```
HOuvreConnexion(MaConnexionpower8)

SI ASsiTraceActive(MaConnexionpower8) ALORS
        sLib est une chaîne
        sLib = ASGetTraceLib(MaConnexionpower8)
        Info(sLib)

        sFic est une chaîne
        sFic = ASGetTraceFic(MaConnexionpower8)
        Info(sFic)

        nLvl est un entier
        nLvl = ASGetTraceLvl(MaConnexionpower8)
        Info(nLvl)
FIN
```

Fonctions Gestion de Travaux

ASJobLogList

Liste les objets d'une joblog AS/400.

Syntaxe

```
Result = ASJobLogList(ASJobLogCollection, Travail, Utilisateur, Numéro du travail,
Nombre de Message, Direction, Connexion)
```

En anglais : `ASJobLogList.`

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

ASJobLogCollection

Variable de type ASJobLogCollection qui va contenir les résultats.

Travail

Filtre selon le nom du travail, valeurs spéciales *, *INT. La valeur * est utilisée si aucune valeur n'est précisée.

Utilisateur

Nom d'un utilisateur spécifique ou blanc quand le nom du travail est la valeur spéciale * ou *INT.

Numéro du travail

Un numéro spécifique de travail ou blanc quand le paramètre du nom du travail est la valeur spécial * ou *INT.

Nombre de Message

Nombre maximal de messages renvoyés.

Direction

Direction de lecture, vous devez utiliser *NEXT ou *PRV. La valeur *NEXT est utilisée si aucune valeur n'est précisée.

Connexion

Connexion - nom de la connexion

Exemple

```
listJobLog est un ASJobLogCollection

//comportement par défaut liste les joblogs
SI PAS ASJobLogList(listJobLog,"","", "","","", MaConnexion1)
 Erreur(ErreurInfo(errComplet))
FIN
```

ASJobList

Liste l'ensemble des jobs actifs d'un sous-système AS/400.

Syntaxe

```
Result = ASJobList(ASJobCollection, Travail, Utilisateur, Numéro du travail, Type,
Statut, Connexion)
```

En anglais : ASJobList.

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

ASJobCollection

Variable de type AsJobCollection qui va contenir les résultats.

Travail

Filtre selon le nom du travail, valeur spéciales *CURRENT, *ALL. La valeur *ALL est utilisée si aucune valeur n'est précisée.

Utilisateur

Filtre selon le nom de l'utilisateur qui a créé le travail, valeur spécial *CURRENT, *ALL. La valeur *CURRENT est utilisée si aucune valeur n'est précisée.

Numéro du travail

Filtre selon un numéro du travail, valeurs spéciales *, *INT. La valeur * est utilisée si aucune valeur n'est précisée.

Type

Filtre selon le type du travail, valeurs spéciales *, A, B, I, M, R, S, W, X. La valeur * est utilisée si aucune valeur n'est précisée.

Statut

Filtre selon le statut du travail, valeurs spéciales *ACTIVE, *JOBQ, *OUTQ, *ALL. La valeur *ACTIVE est utilisée si aucune valeur

Connexion

Connexion - Nom de la connexion.

Exemple

```
listJob est un ASJobCollection

//comportement par défaut liste les jobs actifs du system.
SI PAS ASJobList(listJob,"","", "","","","", MaConnexion1)
 Erreur(ErreurInfo(errComplet))
FIN
```

Objet de type ASJobCollection

Ce type d'objet est utilisé en retour de la fonction ASJobList.

Un ASJobCollection contient un certain nombre d'objet ASJob. Les objets ASJob ont les propriétés suivantes :

Propriété	Description
Job_name	Job name used
Job_user_name	User name used
Job_number	Job number used
Job_internal_id	Internal job identifier
Job_status	Status
Job_type	Job type
Job_subtype	Job subtype
Job_info_status	Job information status
Job_act_job_sts	Active job status

Job_alw_multi_threads	Allow multiple threads
Job_act_endjob_sts	Active job status for jobs ending
Job_brkmsg	Break message handling
Job_cancel_key	Cancel key
Job_ccsid	Coded character set ID
Job_cntryid	Country or region ID
Job_cpu_time	Processing unit time used, if less than 2,147,483,647 milliseconds
Job_usrprf	Current user profile
Job_completion_sts	Completion status
Job_pool_id	Current system pool identifier
Job_char_id_ctrl	Character identifier control
Job_process_unit_time	Processing unit time used - total for the job
Job_process_unit_time_db	Processing unit time used for database - total for the job
Job_datetime_active	Date and time job became active
Job_datetime_in	Date and time job entered system
Job_datetime_sched	Date and time job is scheduled to run
Job_datetime_jobq	Date and time job was put on this job queue
Job_datfmt	Date format
Job_datsep	Date separator
Job_dbcs_cap	DBCS-capable
Job_ddm_handle	DDM conversation handling
Job_dftwait	Default wait
Job_devrcyacn	Device recovery action
Job_devname	Device name
Job_dftccsid	Default coded character set identifier
Job_decfmt	Decimal format
Job_datetime_end	Date and time job ended
Job_endsev	End severity
Job_endsts	End status
Job_exitkey	Exit key
Job_func_name	Function name
Job_func_type	Function type
Job_signed_job	Signed-on job
Job_grpprfname	Group profile name
Job_grpprfname_sup	Group profile name - supplemental
Job_inqmsgrply	Inquiry message reply
Job_account_code	Job accounting code
Job_date	Job date
Job_desc_name	Job description name - qualified
Job_queue_name	Job queue name - qualified
Job_queue_pty	Job queue priority
Job_switches	Job switches

Job_jobmsgqfl	Job message queue full action
Job_jobmsgq_size	Job message queue maximum size
Job_usrid	Job user identity
Job_usrid_setting	Job user identity setting
Job_end_reason	Job end reason
Job_log_pending	Job log pending
Job_type_enhanced	Job type - enhanced
Job_langid	Language ID
Job_loglvl	Logging level
Job_logclpgm	Logging of CL programs
Job_logsev	Logging severity
Job_logtext	Logging text
Job_mode_name	Mode name
Job_max_proc_unit_time	Maximum processing unit time
Job_max_tmp_stg_k	Maximum temporary storage in kilobytes
Job_max_threads	Maximum threads
Job_max_tmp_stg_m	Maximum temporary storage in megabytes
Job_mem_pool_name	Memory pool name
Job_msgrpl	Message reply
Job_interactive_trs	Number of interactive transactions
Job_db_lckwait	Number of database lock waits
Job_mch_lckw	Number of internal machine lock waits
Job_nondb_lckw	Number of nondatabase lock waits
Job_aux_ioreq	Number of auxiliary I/O requests
Job_outq_name	Output queue name - qualified
Job_outq_pty	Output queue priority
Job_prttext	Print text
Job_prtdevname	Printer device name
Job_purge	Purge
Job_prd_retcode	Product return code
Job_prog_retcode	Program return code
Job_pending_sgnset	Pending signal set
Job_process_id	Process ID number
Job_response_time	Response time total
Job_runpty	Run priority (job)
Job_routing_data	Routing data
Job_strseq	Sort sequence table - qualified
Job_sts_msghdl	Status message handling
Job_sts_jobq	Status of job on the job queue
Job_sbmjob	Submitter's job name - qualified
Job_sbmmsgq	Submitter's message queue name - qualified
Job_sbsd	Subsystem description name - qualified

Job_syspoolid	System pool identifier
Job_spclenv	Special environment
Job_sgnblk_mask	Signal blocking mask
Job_sgnsts	Signal status
Job_svrtype	Server type
Job_splfile_action	Spooled file action
Job_timsep	Time separator
Job_timeslice	Time slice
Job_timeslice_end	Time-slice end pool
Job_tmpstgk	Temporary storage used in kilobytes
Job_time_db_lckw	Time spent on database lock waits
Job_time_mch_lckw	Time spent on internal machine lock waits
Job_time_nondb_lckw	Time spent on nondatabase lock waits
Job_threadcnt	Thread count

ASChangeNomJob

Personnaliser le texte de « fonction » affiché dans WRKACTJOB pour le job du sous-système Easycom correspondant à la connexion.

Syntaxe

```
Result = ASChangeNomJob(NouveauNom, Connexion)
```

En anglais : `ASChangeJobName.`

Paramètres

Result

Int – Renvoie -1 en cas d'erreur, sinon 0.

NouveauNom

Nom qui sera affiché dans WRKACTJOB sur l'AS400.

Connexion

Connexion - Nom de la connexion.

Exemple

```
MaConnexion est une Connexion
// Description de la connexion
MaConnexion..Utilisateur = "aura"
MaConnexion..MotDePasse = "aura"
MaConnexion..Serveur = "power8"
MaConnexion..Provider = hAccèsNatifAS400
MaConnexion..InfosEtendues= "<EASYCOM>"+CRLF+"JOBNAME=CED"+CRLF+"</EASYCOM>"
HOuvreConnexion(MaConnexion)

//Si on fait un WRKACTJOB à ce moment-là, on obtient :
```

```
Opt  S-syst/trav   cours      Type   % UC  Fonction        Etat
_    EASYCOM       QSYS       SBS     0,0                  DEQW
_      CED         QPGMR      BCH     0,0  PGM-EASYCOM     TIMW
_      EASYCOMD    QTCP       ASJ     0,0  PGM-EASYCOMD    SELW
_      NISSA01     QPGMR      BCH     0,0  PGM-EASYCOM     TIMW
```

```
SI PAS ASChangeNomJob("TEST", MaConnexion) ALORS
       Info(ErreurInfo())
FIN


//Si on fait un WRKACTJOB après le ASChangeNomJob, on obtient :
```

```
Opt  S-syst/trav   cours      Type   % UC  Fonction        Etat
_    EASYCOM       QSYS       SBS     0,0                  DEQW
_      CED         QPGMR      BCH     0,0  USR-TEST        TIMW
_      EASYCOMD    QTCP       ASJ     0,0  PGM-EASYCOMD    SELW
_      NISSA01     QPGMR      BCH     0,0  PGM-EASYCOM     TIMW
```

```
HFermeConnexion(MaConnexion)
```

ASSoumetBCI

Lancer un travail immédiat (BCI). Attention, si vous voulez passez des données au programme en question, celui-ci doit les recupérer en se basant sur l'exemple founi (EASYCOMXMP/QCLSRC membre TESTPUTRQ).

Il n'est pas possible d'appeler directement un programme avec des paramètres tel qu'avec la commande ASExec.

Syntaxe

```
Result = ASSoumetBCI(NomProgramme, Librairie, Utilisateur, Données, Multi-Thread,
CurrLIBL, Connexion)
```

En anglais : `ASSubmitBCI.`

Paramètres

Result

Chaine de caractères ANSI – ID du job si tout s'est bien passé, vide en cas d'erreur.

NomProgramme

Nom du programme AS400 à appeler en batch.

Librairie

Librairie du programme AS400.

Utilisateur

Nom du profil utilisateur qui créé le travail.

Données

Données à passer en paramètre du travail.

Multi-thread

Booléen – si vrai autorise le travail à fonctionner en mode multithread.

CurrLIBL

Booléen – si vrai, garde la LIBL courante pour l'exécution du travail.

Connexion

Connexion - nom de la connexion

Exemple

```
JobId = ASSoumetBCI(pgm,lib,sUser,Data,bMultiThread,bCurrlibl)
```

ASSoumetPJ

Lancer un travail en pre-start job (PJ). Attention, si vous voulez passez des données au programme en question, celui-ci doit les recupérer en se basant sur l'exemple founi (EASYCOMXMP/QCLSRC membre TESTPUTRQ).

Il n'est pas possible d'appeler directement un programme avec des paramètres tel qu'avec la commande ASExec.

Syntaxe

```
Result = ASSoumetPJ(Program, User, Data, SBS, Connexion)
```

En anglais : `ASSubmitPJ.`

Paramètres

Result

Chaine de caractères ANSI – ID du job si tout c'est bien passé, vide en cas d'erreur.

NomProgramme

Nom du programme AS400 à appeler en batch.

Utilisateur

Nom du profil utilisateur qui créé le travail.

Données

Données à passer en paramètre du travail.

SBS

Sous-système du pre-start job à appeler.

Connexion

Connexion - Nom de la connexion.

Exemple

```
JobId = ASSoumetPJ(pgm,sUser,Data,sbs)
```

ASEtatJob

Permet d'obtenir le statut d'un travail lancé en BIC ou PJ.

Syntaxe

```
Result = ASEtatJob(JobID, Connexion)
```

En anglais : `ASJobStatus.`

Paramètres

Result

Chaine de caractère - Statut du travail.

JobID

Identifiant du job tel que retourné par ASSoumetBIC ou ASSoumetPJ.

Connexion

Connexion - nom de la connexion.

Fonctions Gestion des locks

ASJobRecordLocks

Liste tous les verrous (locks) d'un job AS/400 en cours.

Syntaxe

```
Result = ASJobrecordLocks(ASJobRecordLockCollection, Travail, Utilisateur, Numéro
du travail, Connexion)
```

En anglais : `ASJobrecordLocks.`

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

ASJobRecordLockCollection

Variable de type ASJobRecordLockCollection qui va contenir les résultats.

Travail

Nom du travail.

Utilisateur

Nom d'un utilisateur spécifique.

Numéro du travail

Un numéro spécifique de travail.

Connexion

Connexion - nom de la connexion.

Exemple

```
 lockList est une ASJobRecordLockCollection
 nbLock est un entier
 monLock est un ASJobRecordLock

 ASJobRecordLocks(lockList,jb.Job_name,jb.Job_user_name,jb.Job_number,PrinciCo
nnexion)
 nbLock = lockList.CollectionASJobRecordLock..Occurrence
 POUR i =1 A nbLock
       monLock = lockList.CollectionASJobRecordLock[i]
       TableauAjoute(gtabAsJobRecordLock,monLock)
FIN
```

ASRecordLocks

Liste les verrous (locks) présents sur un fichier de base de données.

Syntaxe

```
Result = ASRecordLocks(ASRecordLockCollection, Fichier, Bibliothèque, Membre,
Connexion)
```

En anglais : `ASRecordLocks.`

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

ASRecordLockCollection

Variable de type ASRecordLockCollection qui va contenir les résultats.

Fichier

Fichier sur lequel on liste les locks.

Bibliothèque

Bibliothèque, il est possible d'utiliser la valeur *LIBL ou *CURLIB.

Membre

Membre concerné, il est possible d'utiliser *FIRST.

Connexion

Connexion - nom de la connexion

Exemple

```
lockList est une ASRecordLockCollection
monObjet est un ASRecordLock

nb est un entier
Sablier(Vrai)
SI ASRecordLocks(lockList,"SP_CUST","*LIBL","*FIRST",PrinciConnexion) ALORS
      nb = lockList.CollectionASRecordLock..Occurrence
      POUR nIndice= 1 A nb
            monObjet = lockList.CollectionASRecordLock[nIndice]
            TableauAjoute(gtabAsRecordLock, monObjet)
      FIN
FIN

TableAffiche(TABLE_recordlocks)
Sablier(Faux)
```

ASObjLocks

Liste les verrous (locks) présents sur un objet.

Syntaxe

```
Result = ASVerrouObjets(ASObjLockCollection, Connexion, NomObjet, NomBibliothèque,
TypeObjet, [NomMembre], [objPathName],[objASPName])
```

En anglais : `ASObjLocks`.

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

ASObjLockCollection

Variable de type ASObjLockCollection qui va contenir les résultats. Collection d'ASObjLock.

Connexion

Connexion - Nom de la connexion.

NomObjet

Chaîne - Nom de l'objet pour lequel on va chercher les verrous.

NomBibliothèque

Chaîne - Nom de la bibliothèque contenant l'objet pour lequel on va chercher les verrous.

Il est possible d'utiliser la valeur *LIBL ou *CURLIB.

TypeObjet

Chaîne – Type de l'objet dont on cherche les verrous. Exemple : *FILE, *LIBL ou *MSGQ.

Voir en annexe la liste complète des types utilisables.

NomMembre

Chaîne – Si applicable, mettre le nom du membre dont on cherche les verrous.

Valeurs possibles :

*ALL, *FIRST, *NONE ou le nom du membre pour TypeObjet=*FILE.

*NONE pour tous les autres types.

[Optionnel] si ce paramètre n'est pas spécifié alors la valeur par défaut utilisée est *NONE.

objPathName

Chaîne – [Optionnel] Chemin physique de l'objet dont on veut spécifiquement connaître les verrous.

Si utilisé le nom de l'objet (**NomObjet**) doit être à *OBJPATH, le nom du membre (**NomMembre)** à *NONE et la bibliothèque (**NomBibliothèque)** et le type (**TypeObjet)** à vide.

objASPName

Chaîne – [Optionnel] Nom de l'ASP (Auxiliary Storage Pool) où rechercher l'objet.

Si utilisé : doit avoir la valeur * si **NomBibliothèque=** *LIBL ou *CURLIB ou si **NomObjet=***OBJPATH.

Si l'objet est une bibliothèque ou un ASP ou encore si *SYSBAS est spécifié, la bibliothèque doit être QSYS.

Remarque

Si NomMembre n'est pas spécifié et que TypeObjet=*FILE, alors on va chercher les verrous sur le fichier lui-même.

Si on veut les verrous sur les membres du fichier, il faut : TypeObjet=*FILE et NomMembre= *ALL.

Exemple

1ère méthode pour parcourir la collection résultat (conseillée):

```
sObjname est une chaîne sur 10;
sObjlibname est une chaîne sur 10;
sObjname = "EASYCOMLOG";
sObjlibname = "EASYCOM";
sFullQualifiedObjName est une chaîne sur 20;
sFullQualifiedObjName = Complète(sObjname,10," ") + Complète(sObjlibname,10," ");
sObjType est une chaîne  = "*FILE";
sMemberName est une chaîne = "*ALL";

sResLockSkel est une chaîne = [
```

```
	Job Name . . . . . . . . . . . . . . . . . . . . . . . . : %1
	Job Username . . . . . . . . . . . . . . . . . . . . . . : %2
	Job Number . . . . . . . . . . . . . . . . . . . . . . . : %3
	Lock State . . . . . . . . . . . . . . . . . . . . . . . : %4
	Lock Status . . . . . . . . . . . . . . . . . . . . . . . : %5
	Lock Status Desc . . . . . . . . . . . . . . . . . . . . : %6
	Lock Type . . . . . . . . . . . . . . . . . . . . . . . . : %7
	Lock Type Desc . . . . . . . . . . . . . . . . . . . . . : %8
	Member Name . . . . . . . . . . . . . . . . . . . . . . . : %9
	Share . . . . . . . . . . . . . . . . . . . . . . . . . . : %10
	Share Desc . . . . . . . . . . . . . . . . . . . . . . . : %11
	Lock Scope . . . . . . . . . . . . . . . . . . . . . . . : %12
	Lock Scope Desc . . . . . . . . . . . . . . . . . . . . . : %13
	Thread Identifier . . . . . . . . . . . . . . . . . . . . : %14
	Lock %15 on %16
]

lockColl est une ASObjLockCollection ;

HOuvreConnexion(MaConnexionpower8)

//SI PAS ASObjLocks(lockColl, MaConnexionpower8,sObjname,
sObjlibname,sObjType,sMemberName,"","*SYSBAS") ALORS
SI PAS ASObjLocks(lockColl, MaConnexionpower8,sObjname, sObjlibname,sObjType) ALORS
//SI PAS ASVerrousObjets(lockColl, MaConnexionpower8,"TSECOFR",
"QUSRSYS","*MSGQ","","","*SYSBAS") ALORS
//SI PAS ASVerrousObjets(lockColl, MaConnexionpower8,"QCQMONMQ",
"QSVMSS","*MSGQ","","","*") ALORS
	Erreur(ErreurInfo(errComplet))
SINON
	nInd est un entier
	nInd = 1
	POUR TOUT objlock DE lockColl

		sResLock est une chaîne;
		sResLock = ChaîneConstruit(sResLockSkel,
		objlock.JobName,
		objlock.jobUsername,
		objlock.JobNumber,
		objlock.LockState,
		objlock.LockStatus,
		objlock.lockStatusDescription,
		objlock.locktype,
		objlock.locktypeDescription,
		objlock.MemberName,
		objlock.share,
		objlock.shareDescription,
		objlock.LockScope,
		objlock.lockScopeDescription,
		objlock.threadId,
		nInd,
		lockColl.CollectionASObjLock..Occurrence)
		Info(sResLock);
		nInd++

	FIN

FIN
```

2ème méthode pour parcourir la collection résultat :

```
//*** Récupéré lock objet ***/
sObjname est une chaîne sur 10;
sObjlibname est une chaîne sur 10;
sObjname = "EASYCOMLOG";
sObjlibname = "EASYCOM";
sFullQualifiedObjName est une chaîne sur 20;
sFullQualifiedObjName = Complète(sObjname,10," ") + Complète(sObjlibname,10," ");
sObjType est une chaîne  = "*FILE";
sMemberName est une chaîne = "*ALL";


sResLockSkel est une chaîne = [
       Job Name . . . . . . . . . . . . . . . . . . .    : %1
       Job Username . . . . . . . . . . . . . . . . .    : %2
       Job Number . . . . . . . . . . . . . . . . . .    : %3
       Lock State . . . . . . . . . . . . . . . . . .    : %4
       Lock Status . . . . . . . . . . . . . . . . . .   : %5
       Lock Status Desc . . . . . . . . . . . . . . .    : %6
       Lock Type . . . . . . . . . . . . . . . . . . .   : %7
       Lock Type Desc . . . . . . . . . . . . . . . .    : %8
       Member Name . . . . . . . . . . . . . . . . . .   : %9
       Share . . . . . . . . . . . . . . . . . . . . .   : %10
       Share Desc . . . . . . . . . . . . . . . . . .    : %11
       Lock Scope . . . . . . . . . . . . . . . . . .    : %12
       Lock Scope Desc . . . . . . . . . . . . . . . .   : %13
       Thread Identifier . . . . . . . . . . . . . . .   : %14
       Lock %15 on %16
]

lockColl est une ASObjLockCollection ;

HOuvreConnexion(MaConnexionpower8)

SI PAS ASObjLocks(lockColl, MaConnexionpower8,sObjname,
sObjlibname,sObjType,sMemberName,"","*SYSBAS") ALORS
//SI PAS ASObjLocks(lockColl, MaConnexionpower8,sObjname, sObjlibname,sObjType) ALORS
//SI PAS ASVerrousObjets(lockColl, MaConnexionpower8,"TSECOFR",
"QUSRSYS","*MSGQ","","","*SYSBAS") ALORS
//SI PAS ASVerrousObjets(lockColl, MaConnexionpower8,"QCQMONMQ",
"QSVMSS","*MSGQ","","","*") ALORS
       Erreur(ErreurInfo(errComplet))
SINON

       POUR I = 1 _À_ lockColl.CollectionASObjLock..Occurrence
              sResLock est une chaîne;
              objlock est un ASObjLock;
              objlock = lockColl[I];
              sResLock = ChaîneConstruit(sResLockSkel,
              objlock.JobName,
              objlock.jobUsername,
              objlock.JobNumber,
              objlock.LockState,
              objlock.LockStatus,
              objlock.lockStatusDescription,
              objlock.locktype,
              objlock.locktypeDescription,
              objlock.MemberName,
              objlock.share,
              objlock.shareDescription,
              objlock.LockScope,
              objlock.lockScopeDescription,
              objlock.threadId,
```

```
                I,
                lockColl.CollectionASObjLock..Occurrence)
                Info(sResLock);

        FIN

FIN
```

Objet de type ASObjLockCollection

Ce type d'objet est utilisé en retour de la fonction `ASObjLocks`. Il s'agit d'une collection d'objets **AsObjLock**.

Propriétés de l'objet AsObjLock

Nom propriété en Français	Nom propriété en Anglais	Type
jobname	jobname	chaîne
jobUsername	jobUsername	chaîne
jobNumber	jobNumber	chaîne
lockState	lockState	chaîne
lockStatus	lockStatus	chaîne
lockStatusDescription	lockStatusDescription	chaîne
locktype	locktype	chaîne
locktypeDescription	locktypeDescription	chaîne
memberName	memberName	chaîne
share	share	chaîne
shareDescription	shareDescription	chaîne
lockScope	lockScope	chaîne
lockScopeDescription	lockScopeDescription	chaîne
threadId	threadId	chaîne

Remarque

Pour les propriétés suivantes : **lockStatus**, **locktype**, **shareDescription** et **lockScope**, on renvoie l'information brute et une description supplémentaire :

LockStatus :

- "The lock is currently held by the job or thread."
- "The job or thread is waiting for the lock (synchronous)."
- "The job or thread has a lock request outstanding for the object (asynchronous)."

LockType:

- "Lock on the object"
- "Lock on the member control block"
- "Lock on the access path used to access a member's data"
- "Lock on the actual data within the member"

Share Decription:

- "The file is not shared, the file is a physical file, or the field is not applicable to object type."
- "The file is shared."

Lock Scopes :

- "Job Scope"
- "Thread Scope"
- "Lock Space Scope"

Fonctions Output Queues/Remote Output Queues/Writers(Editeurs)

ASOutputQueueList

Liste toutes les files de sortie (Output Queues *OUTQ).

Syntaxe

```
Result = ASOutputQueueList(ASOutputQueueCollection, Connexion,
[FiltreDistant,NomOutputQueue])
```

En anglais : `ASOutputQueueList.`

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

La fonction ErreurInfo permet d'identifier l'erreur.

ASOutputQueueCollection

Collection de files de sorties (Output Queues *OUTQ) qui va contenir les résultats.

Connexion

Connexion - nom de la connexion.

FiltreDistant

Optionnel. Valeur parmi les constantes suivantes :

ASOutQRemoteOnly : Filtre uniquement les *OUTQ distantes (Remote Output Queues)

ASOutQLocalOnly : Filtre uniquement les *OUTQ locales

ASOutQBothRemoteAndLocal : Pas de filtre, correspond à la combinaison ASOutQRemoteOnly+ASOutQLocalOnly

NomOutputQueue

Chaîne de caractère (avec guillemets) - Nom du ou des file(s) de sortie recherchée(s). Optionnel.

Si vous ne voulez pas filtrer par nom : laissez vide ou mettez "*".

Si vous voulez faire une recherche approximative (commençant par) finissez par * (ex: "PRT*").

Exemple

```
gTest est un ASOutputQueueCollection


SI PAS ASOutputQueueList(gTest,MaConnexion1) ALORS
//SI PAS ASOutputQueueList(gTest,MaConnexion1,ASOutQBothRemoteAndLocal) ALORS
//SI PAS ASOutputQueueList(gTest,MaConnexion1,ASOutQRemoteOnly) ALORS
//SI PAS ASOutputQueueList(gTest,MaConnexion1,ASOutQLocalOnly) ALORS
//SI PAS ASOutputQueueList(gTest,MaConnexion1,ASOutQBothRemoteAndLocal,"HP*") ALORS
//SI PAS ASOutputQueueList(gTest,MaConnexion1,ASOutQRemoteOnly,"HP*") ALORS
Info(ErreurInfo(errComplet))
```

```
FIN
```

ASOutputQueueClear

Vide une file de sortie (Output Queues *OUTQ).

Syntaxe

```
Result = ASOutputQueueClear(ASOutputQueue, Connexion)
```

En anglais : `ASOutputQueueClear.`

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

La fonction ErreurInfo permet d'identifier l'erreur.

ASOutputQueue

File de sortie (Output Queues *OUTQ).

Connexion

Connexion - nom de la connexion

Exemple

```
gTest est un ASOutputQueueCollection
nResSelect est un entier  = TableSelect(TABLE_CollectionASOutputQueue)
asoq est une ASOutputQueue = gTest.CollectionASOutputQueue[nResSelect]
//1 : &Oui
//2 : &Annuler
SELON Dialogue("Êtes-vous sûr(e) de vouloir mettre cette file d'attente à blanc?")
	// &Oui
	CAS 1
		SI PAS ASOutputQueueClear(asoq, MaConnexion1) ALORS
			Info(ErreurInfo(errComplet))
		FIN
		Proc_Gestion_ASOutQ(nResSelect)
	// &Annuler
	CAS 2
		RETOUR
FIN
```

ASOutputQueueRelease

Suspend une file de sortie (Output Queues *OUTQ).

Syntaxe

```
Result = ASOutputQueueRelease(ASOutputQueue, Connexion)
```

En anglais : `ASOutputQueueRelease.`

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

La fonction ErreurInfo permet d'identifier l'erreur.

ASOutputQueue

File de sortie (Output Queues *OUTQ).

Connexion

Connexion - nom de la connexion

Exemple

```
gTest est un ASOutputQueueCollection
nResSelect est un entier  = TableSelect(TABLE_CollectionASOutputQueue)
asoq est une ASOutputQueue = gTest.CollectionASOutputQueue[nResSelect]
SI PAS ASOutputQueueRelease(asoq, MaConnexion1) ALORS
       Info(ErreurInfo(errComplet))
FIN
```

ASOutputQueueHold

Tient une file de sortie (Output Queues *OUTQ).

Syntaxe

```
Result = ASOutputQueueHold(ASOutputQueue, Connexion)
```

En anglais : `ASOutputQueueHold.`

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

La fonction ErreurInfo permet d'identifier l'erreur.

ASOutputQueue

File de sortie (Output Queues *OUTQ).

Connexion

Connexion - nom de la connexion

Exemple

```
gTest est un ASOutputQueueCollection
nResSelect est un entier  = TableSelect(TABLE_CollectionASOutputQueue)
asoq est une ASOutputQueue = gTest.CollectionASOutputQueue[nResSelect]
SI PAS ASOutputQueueHold(asoq, MaConnexion1) ALORS
       Info(ErreurInfo(errComplet))
FIN
```

ASWriterInfo

Permet de récupérer les infos sur un éditeur.

Syntaxe

```
Result = ASWriterInfo(ASWriterDetailed, Connexion, nomImprimante, nomEditeur)
```

En anglais : `ASWriterInfo.`

Paramètres

Result
Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.
La fonction ErreurInfo permet d'identifier l'erreur.

ASWriterDetailed
Editeur détaillé.

Connexion
Connexion - nom de la connexion

nomImprimante
Chaîne de caractère (avec guillemets) - Nom de l'imprimante dont on veut les informations de l'éditeur.

nomEditeur
Chaîne de caractère (avec guillemets) - Nom de l'éditeur dont on veut les informations.

Exemple

```
aswtr est un ASWriterDetailed
SI PAS ASWriterInfo(aswtr, MaConnexion1,"HP","HP") ALORS
       Info(ErreurInfo(errComplet))
FIN
```

ASWriterStart

Permet de démarrer un éditeur. Equivalent à STRPRTWTR.

Syntaxe 1

```
Result = ASWriterStart(ASWriter, ASOutputQueue, Connexion)
```

En anglais : `ASWriterStart.`

Paramètres

Result
Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.
La fonction ErreurInfo permet d'identifier l'erreur.

ASWriter
Editeur.

ASOutputQueue
File de sortie (Ouput Queue *OUTQ).

Connexion
Connexion - nom de la connexion

Syntaxe 2

```
Result = ASWriterStart(ASWriterDetailed, Connexion)
```

En anglais : `ASWriterStart.`

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

La fonction ErreurInfo permet d'identifier l'erreur.

ASWriterDetailed

Editeur détaillé.

Connexion

Connexion - nom de la connexion

Exemple

```
gTest est un ASOutputQueueCollection
nResSelectedQ est un entier  = TableSelect(TABLE_CollectionASOutputQueue)
nResSelectedWtr est un entier  = TableSelect(TABLE_CollectionASWriter)
asoq est une ASOutputQueue = gTest[nResSelectedQ]
aswtr est un ASWriterDetailed = asoq.WriterCollection[nResSelectedWtr]
SI PAS ASWriterStart(aswtr, MaConnexion1) ALORS
       Info(ErreurInfo(errComplet))
FIN
```

ASWriterEnd

Arrête un éditeur (fonctionne avec un éditeur distant également). Equivalent à un ENDWTR.

Syntaxe 1

```
Result = ASWriterEnd(ASWriter, Connexion,[stopOption])
```

En anglais : `ASWriterEnd.`

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

La fonction ErreurInfo permet d'identifier l'erreur.

ASWriter

Editeur.

Connexion

Connexion - nom de la connexion

StopOption

Optionnel. Valeur parmi les constantes suivantes :

ASCntld: *CNTLD. Valeur par défaut.

ASImmed: *IMMED

ASPageEnd: *PAGEEND

Syntaxe 2

```
Result = ASWriterEnd(ASWriterDetailed, Connexion,[stopOption])
```

En anglais : `ASWriterEnd.`

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

La fonction ErreurInfo permet d'identifier l'erreur.

ASWriterDetailed

Editeur détaillé.

Connexion

Connexion - nom de la connexion

StopOption

Optionnel. Valeur parmi les constantes suivantes :

ASCntld: *CNTLD. Valeur par défaut.

ASImmed: *IMMED

ASPageEnd: *PAGEEND

Exemple

```
gTest est un ASOutputQueueCollection

nResSelectedQ est un entier  = TableSelect(TABLE_CollectionASOutputQueue)
nResSelectedWtr est un entier  = TableSelect(TABLE_CollectionASWriter)

aswtr est un ASWriterDetailed =
gTest[nResSelectedQ].WriterCollection[nResSelectedWtr]

//SI PAS ASWriterEnd(aswtr, MaConnexion1, ASCntld) ALORS
//SI PAS ASWriterEnd(aswtr, MaConnexion1) ALORS
//SI PAS ASWriterEnd(aswtr, MaConnexion1, ASPageEnd) ALORS
SI PAS ASWriterEnd(aswtr, MaConnexion1, ASImmed) ALORS
       Info(ErreurInfo(errComplet))
FIN
```

ASWriterRestart

Redémarre un éditeur. Equivalent a un ENDWTR puis à une vérification d'état jusqu'à preuve d'arrêt puis un STRPRTWTR.

Syntaxe 1

```
Result = ASWriterRestart(ASWriter, ASOutputQueue, Connexion,[stopOption])
```

En anglais : `ASWriterRestart.`

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

La fonction ErreurInfo permet d'identifier l'erreur.

ASWriter

Editeur.

ASOutputQueue

File de sortie (Ouput Queue *OUTQ).

Connexion

Connexion - nom de la connexion

StopOption

Optionnel. Valeur parmi les constantes suivantes :

ASCntld: *CNTLD. Valeur par défaut.

ASImmed: *IMMED

ASPageEnd: *PAGEEND

Syntaxe 2

```
Result = ASWriterRestart(ASWriterDetailed, Connexion,[stopOption])
```

En anglais : `ASWriterRestart.`

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

La fonction ErreurInfo permet d'identifier l'erreur.

ASWriterDetailed

Editeur détaillé.

Connexion

Connexion - nom de la connexion

StopOption

Optionnel. Valeur parmi les constantes suivantes :

ASCntld: *CNTLD. Valeur par défaut.

ASImmed: *IMMED

ASPageEnd: *PAGEEND

Exemple

```
gTest est un ASOutputQueueCollection

nResSelectedQ est un entier  = TableSelect(TABLE_CollectionASOutputQueue)
nResSelectedWtr est un entier  = TableSelect(TABLE_CollectionASWriter)
asoq est une ASOutputQueue = gTest[nResSelectedQ]
aswtr est un ASWriterDetailed = asoq.WriterCollection[nResSelectedWtr]
SI PAS ASWriterRestart(aswtr, MaConnexion1) ALORS
       Info(ErreurInfo(errComplet))
FIN
```

ASWriterStartRemote

Permet de démarrer un éditeur distant.

Syntaxe 1

```
Result = ASWriterStartRemote(ASWriter, ASOutputQueue, Connexion)
```

En anglais : ASWriterStartRemote.

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

La fonction ErreurInfo permet d'identifier l'erreur.

ASWriter

Editeur.

ASOutputQueue

File de sortie (Ouput Queue *OUTQ).

Connexion

Connexion - nom de la connexion

Syntaxe 2

```
Result = ASWriterStartRemote(ASWriterDetailed, Connexion)
```

En anglais : ASWriterStartRemote.

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

La fonction ErreurInfo permet d'identifier l'erreur.

ASWriterDetailed

Editeur détaillé.

Connexion

Connexion - nom de la connexion

Exemple

```
gTest est un ASOutputQueueCollection

nResSelectedQ est un entier  = TableSelect(TABLE_CollectionASOutputQueue)
nResSelectedWtr est un entier  = TableSelect(TABLE_CollectionASWriter)
asoq est une ASOutputQueue = gTest[nResSelectedQ]

aswtr est un ASWriterDetailed = asoq.WriterCollection[nResSelectedWtr]

SI PAS ASWriterStartRemote(aswtr, MaConnexion1) ALORS
       Info(ErreurInfo(errComplet))
FIN
```

ASWriterRestartRemote

Redémarre un éditeur distant.
A utiliser si l'on souhaite redémarrer un éditeur distant car si on l'arrête on ne pourra pas le redémarrer.

Syntaxe 1

```
Result = ASWriterRestartRemote(ASWriter, ASOutputQueue, Connexion,[stopOption])
```

En anglais : ASWriterRestartRemote.

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

La fonction ErreurInfo permet d'identifier l'erreur.

ASWriter

Editeur.

ASOutputQueue

File de sortie (Ouput Queue *OUTQ).

Connexion

Connexion - nom de la connexion

StopOption

Optionnel. Valeur parmi les constantes suivantes :

ASCntld: *CNTLD. Valeur par défaut.

ASImmed: *IMMED

ASPageEnd: *PAGEEND

Syntaxe 2

```
Result = ASWriterRestartRemote(ASWriterDetailed, Connexion,[stopOption])
```

En anglais : ASWriterRestartRemote.

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

La fonction ErreurInfo permet d'identifier l'erreur.

ASWriterDetailed

Editeur détaillé.

Connexion

Connexion - nom de la connexion

StopOption

Optionnel. Valeur parmi les constantes suivantes :

ASCntld: *CNTLD. Valeur par défaut.

ASImmed: *IMMED

ASPageEnd: *PAGEEND

Exemple

```
gTest est un ASOutputQueueCollection

nResSelectedQ est un entier  = TableSelect(TABLE_CollectionASOutputQueue)
nResSelectedWtr est un entier  = TableSelect(TABLE_CollectionASWriter)
asoq est une ASOutputQueue = gTest[nResSelectedQ]
aswtr est un ASWriterDetailed = asoq.WriterCollection[nResSelectedWtr]
SI PAS ASWriterRestartRemote(aswtr, MaConnexion1,ASImmed) ALORS
       Info(ErreurInfo(errComplet))
FIN
```

Objet de type ASOutputQueueCollection

Ce type d'objet est utilisé en retour de la fonction `ASOutputQueueList` . Il s'agit d'une collection d'objets **ASOutputQueue.**

Propriétés de l'objet ASOutputQueue

Nom propriété	Description	Type AS400
OutputQueueName	Output queue name	CHAR(10)
OutputQueueLibName	Output queue library name	CHAR(10)
fileOrder	Order of files on queue	CHAR(10)
displayAnyFile	Display any file	CHAR(10)
jobSeparator	Job separators	BINARY(4)
operatorControlled	Operator controlled	CHAR(10)
dataQName	Data queue name	CHAR(10)
dataQLibName	Data queue library name	CHAR(10)
AuthToCheck	Authority to check	CHAR(10)
NbOfFiles	Number of files	BINARY(4)
OutputQueueStatus	Output queue status	CHAR(10)
TextDescription	Text description	CHAR(50)
NbSpoolFilePageSpec	Number of spooled file pages specified	BINARY(4)
NbWritersStartedToQ	Number of writers started to queue	BINARY(4)
NbWritersToAutoStart	Writers to autostart	BINARY(4)
RemoteSysNameType	Remote system name type	CHAR(1)
RemoteSysName	Remote system name	CHAR(255)
RemotePrinterQ	Remote printer queue	CHAR(128)
MsgQName	Message queue name	CHAR(10)
MsgQLibName	Message queue library name	CHAR(10)
ConnectionType	Connection type	BINARY(4)
DestinationType	Destination type	BINARY(4)
VMMVSClass	VM/MVS class	CHAR(1)
FormControlBuff	Forms control buffer	CHAR(8)
HostPrintTransform	Host print transform	CHAR(1)
ManufacturerTypeAndModel	Manufacturer type and model	CHAR(17)
WorkstationCustomObjName	Workstation customizing object name	CHAR(10)
WorkstationCustomObjLibName	Workstation customizing object library name	CHAR(10)
SpoolASPAttribute	Spooled file auxiliary storage pool attribute	CHAR(1)
NbPageEtriesReturned	Number of page entries returned	BINARY(4)
DestinationOptions	Destination options	CHAR(128)
WriterTypeStartedQueue	Writer type started to	CHAR(1)

	queue	
PrintSeparatorPage	Print separator page	CHAR(1)
LongRemotePrinterQueue	Long remote printer queue	CHAR(255)
ImgConf	Image configuration	CHAR(10)
ImgConfLib	Image configuration library	CHAR(10)
NetDirPublishingStatus	Network directory publishing status	CHAR(1)
SpooledFileAspId	Spooled file auxiliary storage pool ID	BINARY(4)
SpooledFileAspDeviceName	Spooled file auxiliary storage pool device name	CHAR(10)
SplFileMaxPageEntries	Spooled file maximum page entries	CHAR(*)

Objet de type ASWriterCollection

Il s'agit d'une collection d'objets **ASWriter.**

Propriétés de l'objet ASWriter

Nom propriété	Description	Type AS400
WriterJobName	Writer job name	CHAR(10)
WriterUserName	Writer job user name	CHAR(10)
WriterJobNumber	Writer job number	CHAR(6)
WriterJobStatus	Writer job status	CHAR(10)
PrinterDeviceName	Printer device name	CHAR(10)

Objet de type ASWriterDetailedCollection

Il s'agit d'une collection d'objets **ASWriterDetailed.**

Propriétés de l'objet ASWriterDetailed

Nom propriété	Description	Type AS400
startedbyUser	Started by user	CHAR(10)
writingStatus	Writing status	CHAR(1)
waitingForMsgStatus	Waiting for message status	CHAR(1)
heldStatus	Held status	CHAR(1)
endPendingStatus	End pending status	CHAR(1)
holdPendingStatus	Hold pending status	CHAR(1)
betweenFileStatus	Between file status	CHAR(1)
betweenCopiesStatus	Between copies status	CHAR(1)
waitingForDataStatus	Waiting for data status	CHAR(1)
waitingForDeviceStatus	Waiting for device status	CHAR(1)
onJobQueueStatus	On job queue status	CHAR(1)
typeOfWriter	Type of writer	CHAR(1)

writerJobName	Writer job name	CHAR(10)
writerJobUserName	Writer job user name	CHAR(10)
writerJobNumber	Writer job number	CHAR(6)
printerDeviceType	Printer device type	CHAR(10)
nbOfSeparator	Number of separators	BINARY(4)
drawerForSeparators	Drawer for separators	BINARY(4)
alignForms	Align forms	CHAR(10)
outputQueueName	Output queue name	CHAR(10)
outputQueueLibName	Output queue library name	CHAR(10)
outputQueueStatus	Output queue status	CHAR(1)
formType	Form type	CHAR(10)
msgOption	Message option	CHAR(10)
autoEndWriter	Automatically end writer	CHAR(10)
allowDirectPrinting	Allow direct printing	CHAR(10)
msgQName	Message queue name	CHAR(10)
msgQLibName	Message queue library name	CHAR(10)
changesTakeEffect	Changes take effect	CHAR(10)
nextOuputQueueName	Next ouput queue name	CHAR(10)
nextOuputQueueLibName	Next ouput queue library name	CHAR(10)
nextFormtype	Next form type	CHAR(10)
nextMessageOption	Next message option	CHAR(10)
nextFileSeparators	Next file separators	BINARY(4)
nextSeparatorDrawers	Next separator drawer	BINARY(4)
spoolFileName	Spooled file name	CHAR(10)
jobName	Job name	CHAR(10)
userName	User name	CHAR(10)
jobNumber	Job number	CHAR(6)
spoolFileNumber	Spooled file number	BINARY(4)
pageBeingWritten	Page being written	BINARY(4)
totalPages	Total pages	BINARY(4)
copiesLeftToProduce	Copies left to produce	BINARY(4)
totalCopies	Total copies	BINARY(4)
messageKey	Message key	CHAR(4)
initPrinter	Initialize printer	CHAR(1)
printerDeviceName	Printer device name	CHAR(10)
jobSystemName	Job system name	CHAR(8)
spoolFileCreateDate	Spooled file create date	CHAR(7)
spoolFileCreateTime	Spooled file create time	CHAR(6)

Fonctions Gestions de fichiers

ASOpenFileInfoList

Permet de générer une liste d'objets *FILE qui sont actuellement ouvert dans la connexion (JOB courant), dans un JOB spécifique, ou dans le thread du JOB spécifié en paramètre.

Syntaxe

```
Result = ASOpenFileInfoList(ASOpenFileInfoCollection, Connexion,
[identifiantJobQualifié, identifiantThread])
```

En anglais : `ASOpenFileInfoList.`

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

La fonction ErreurInfo permet d'identifier l'erreur.

ASOpenFileInfoCollection

Collection d'ASOpenFileInfo qui va contenir les résultats pour chaque fichier.

Connexion

Connexion - nom de la connexion.

identifiantJobQualifié

Optionnel. Chaîne de caractère (avec guillemets).
Nom du JOB qualifié (3 éléments): jobNumber /username /jobname.

identifiantThread

Optionnel. Numéro du thread (entier) du JOB spécifié dans **identifiantJobQualifié.**

Exemple

```
// JOB courant
asopnfilecollec est une ASOpenFileInfoCollection

SI PAS ASOpenFileInfoList(asopnfilecollec,MaConnexion) ALORS
    Info(ErreurInfo(errComplet))
SINON
    I est un entier  = 1
    POUR TOUT asopnf DE asopnfilecollec
        Trace(I,": ",asopnf.fileName, asopnf.threadId)
        I = I+1
    FIN
FIN


// JOB spécifique
asopnfilecollec2 est une ASOpenFileInfoCollection

SI PAS ASOpenFileInfoList(asopnfilecollec2,MaConnexion, "653011/QSYS/QINTER") ALORS
    Info(ErreurInfo(errComplet))
SINON
    J est un entier  = 1
    POUR TOUT asopnf2 DE asopnfilecollec2
        Trace(J,": ",asopnf2.fileName + " FROM " + asopnf2.mbrOrDeviceName,
asopnf2.threadId)
        J = J+1
    FIN
FIN


// Thread d'un JOB spécifique
asopnfilecollec3 est une ASOpenFileInfoCollection

SI PAS ASOpenFileInfoList(asopnfilecollec3,MaConnexion, "653011/QSYS/QINTER",
```

```
Val("2")) ALORS
    Info(ErreurInfo(errComplet))
SINON
    K est un entier  = 1
    POUR TOUT asopnf3 DE asopnfilecollec3
        Trace(K,": ",asopnf3.fileName, asopnf3.threadId)
        K = K+1
    FIN
FIN
```

Objet de type ASOpenFileInfoCollection

Ce type d'objet est utilisé en retour de la fonction `ASOpenFileInfoList` . Il s'agit d'une collection d'objets **ASOpenFileInfo.**

Propriétés de l'objet ASOpenFileInfo

Nom propriété	Description	Type AS400
fileName	File name	CHAR(10)
fileLib	File library	CHAR(10)
mbrOrDeviceName	Member or device name	CHAR(10)
fileType	File type	CHAR(10)
recordFormat	Record format	CHAR(10)
activationGrpName	Activation group name	CHAR(10)
threadId	Thread identifier	CHAR(8)
openOption	Open option	CHAR(1)
activationGrpNumber	Activation group number	BINARY(8)
writeCount	Write count	BINARY(8)
readCount	Read count	BINARY(8)
readWriteCount	Write/read count	BINARY(8)
otherIoCount	Other I/O count	BINARY(8)
rrn	Relative record number	BINARY(8)
nbSharedOpens	Number of shared opens	BINARY(8)
objAuxStoragePoolNumber	Object auxiliary storage pool	BINARY(4)

	number	
libAuxStoragePoolNumber	Library auxiliary storage pool number	BINARY(4)
objAuxStoragePoolName	Object auxiliary storage pool name	CHAR(10)
libAuxStoragePoolName	Library auxiliary storage pool name	CHAR(10)

Fonctions Gestion des Messages d'info ou d'erreur AS/400 (CPA, CPF, etc.)

ASMessageReceive

Permet de récupérer un message d'info ou d'erreur dans une file de message via sa clef (Rempli un ASMessage).

Syntaxe

```
Result = ASMessageReceive(ASMessage, NomMessageQueue, LibMessageQueue, ClefMessage, Connexion)
```

En anglais : `ASMessageReceive.`

Paramètres

Result

Booléen - Vrai si l'opération a été réalisée, Faux en cas de problème.

La fonction ErreurInfo permet d'identifier l'erreur.

ASMessage

Message AS/400 - IBMi.

NomMessageQueue

Chaîne de caractère (avec guillemets) - Nom de la file de message du message *MSGQ.

LibMessageQueue

Chaîne de caractère (avec guillemets) - Bibliothèque de la file de message du message *MSGQ.

ClefMessage

Entier - ID (clef) du message.

Connexion

Connexion - nom de la connexion.

Exemple

```
asoq est une ASOutputQueue = gTest[nResSelectedQ]
aswtr est un ASWriterDetailed = asoq.WriterCollection[nResSelectedWtr]
SI aswtr.messageKey <> 0 ET aswtr.messageKey <> -1 ALORS
	asmsg est un ASMessage

	SI PAS ASMessageReceive(asmsg, aswtr.msgQName, aswtr.msgQLibName,
	aswtr.messageKey,MaConnexion1) ALORS
			Info(ErreurInfo(errComplet))
			RETOUR
	FIN
```

ASMessageRetrieve

Permet de retrouver un squelette de message AS/400 avec ses paramètres, leurs tailles et leurs types (Rempli un ASMessageHelp).

Syntaxe 1

```
Result = ASMessageRetrieve(ASMessageHelp, CodeMessage, Connexion)
```

Syntaxe 2

```
Result = ASMessageRetrieve(ASMessageHelp, CodeMessage, FichierMessage, LibMessage,
Connexion)
```

Syntaxe 3

```
Result = ASMessageRetrieve(ASMessageHelp, CodeMessage, FichierMessage, LibMessage,
afficheCaractèresFormattage, Connexion)
```

En anglais : `ASMessageRetrieve.`

Paramètres

Result

Booléen - Vrai si l'opération a été réalisée, Faux en cas de problème.

La fonction ErreurInfo permet d'identifier l'erreur.

ASMessageHelp

Squelette de message AS/400 - IBMi.

CodeMessage

Chaîne de caractère (avec guillemets) - Code identificateur du message (Ex : CPF1234).

FichierMessage

Chaîne de caractère (avec guillemets) – Nom du fichier contenant le message.

LibMessage

Chaîne de caractère (avec guillemets) – Bibliothèque du fichier contenant le message.

Connexion

Connexion - nom de la connexion à une base de données AS/400.

Exemple

```
asmsgh est un ASMessageHelp
SI PAS ASMessageRetrieve(asmsgh, "CPF1124", MaConnexion1) ALORS
       Info(ErreurInfo(errComplet))
RETOUR
```

ASMessageReply

Permet de répondre à un message reçu via sa file de message et sa clef (ASMessage reçu via ASMessageReceive) avec possibilité d'avoir récupéré une liste des réponses possibles, la réponse par défaut, etc. via ASMessageHelp.

Syntaxe

```
Result = ASMessageReply(ASMessage, Réponse, NomMessageQueue, LibMessageQueue,
ClefMessage, retirerMessage, Connexion)
```

En anglais : `ASMessageReply.`

Paramètres

Result

Booléen - Vrai si l'opération a été réalisée, Faux en cas de problème.

La fonction ErreurInfo permet d'identifier l'erreur.

ASMessage

Message AS/400 - IBMi.

Réponse

Chaîne de caractère (avec guillemets) - Réponse au message.

NomMessageQueue

Chaîne de caractère (avec guillemets) - Nom de la file de message du message *MSGQ.

LibMessageQueue

Chaîne de caractère (avec guillemets) - Bibliothèque de la file de message du message.

ClefMessage

Entier - ID (clef) du message.

retirerMessage

Booléen - VRAI si le message doit être retiré de la file de message après la réponse, FAUX sinon.

Connexion

Connexion - nom de la connexion.

Exemple

```
gAsmsg est un ASMessage

SI PAS ASMessageReply(gAsmsg, reply, "QSYSOPR","QSYS", gAsmsg.messageKey, Faux,
MaConnexion1) ALORS
        Info(ErreurInfo(errComplet))
SINON
```

ASMessageHistoryLog

Permet de lister et filtrer l'historique de tout message historisé sur l'AS/400.

Syntaxe 1

```
Result = ASMessageHistoryLog(ASMessageHistoryLogCollection, Connexion)
```

Syntaxe 2

```
Result = ASMessageHistoryLog(ASMessageHistoryLogCollection,
ASMessageHistoryLogFilter, Connexion)
```

En anglais : `ASMessageHistoryLog.`

Paramètres

Result

Booléen - Vrai si l'opération a été réalisée, Faux en cas de problème.

La fonction ErreurInfo permet d'identifier l'erreur.

ASMessageHistoryLogCollection

Collection d'ASMessageHistoryLog.

ASMessageHistoryLogFilter

Objet permettant de filtrer les logs à retourner.

Connexion

Connexion - nom de la connexion à une base de données AS/400.

Exemple

```
asmsghlc est un ASMessageHistoryLogCollection
SI PAS ASMessageHistoryLog(asmsghlc, MaConnexion1) ALORS
       Info(ErreurInfo(errComplet))
RETOUR
```

ASErreurAide - ASErreurDonnee

En plus des fonctions `ErreurInfo` et `HErreurInfo`, voir rubrique sur la gestion des erreurs, il existe deux fonctions complémentaires qui permettent de remonter le détail complet d'une erreur AS400 de type CPF.

Ces fonctions remontent les informations qui seraient affichées par un DSPMSGD.

ASErreurAide renvoie le texte d'aide complet de la dernière erreur rencontrée.

ASErreurDonnee permet d'extraire les données de la dernière erreur, en donnant une position et une longueur.

Pour connaître les positions et longueurs des variables de messages, utilisez la commande DSPMSGD sur un terminal.

Syntaxe

```
Result = ASErreurAide([Connexion])
Result = ASErreurDonnee(Position,Longueur [,Connexion])
```

En anglais : `ASErrorHelp` et `ASErrorData`

Paramètres

Result

Chaîne de caractères. Résultat de la récupération.

Position

Entier - position de la variable dans la chaîne

Longueur

Entier - longueur de la variable

Connexion (optionnel)

Connexion - Nom de la connexion

Exemple

Récupérer le nom de la contrainte en cas d'erreur d'intégrité.

```
s1 est chaînes
contr_nom est chaîne
```

```
contr_parfic, contr_parlib est chaîne
contr_fic, contr_lib est chaîne

s1 = HErreurInfo(hErrMessage)
SI ExtraitChaîne(s1, 6, RC) = "Message: CPF503A" ALORS
  // Nom de la contrainte
  contr_nom = SansEspace(ASErreurDonnee(176, 258))
  contr_parfic = SansEspace(ASErreurDonnee(448, 10))
  contr_parlib = SansEspace(ASErreurDonnee(458, 10))
  contr_fic = SansEspace(ASErreurDonnee(10, 10))
  contr_lib = SansEspace(ASErreurDonnee(20, 10))
      Info("Erreur intégrité sur contrainte: "+contr_nom+RC+"Fichier parent:
      "+contr_parlib+"/"+contr_parfic+RC+"Fichier de base: "+contr_lib+"/"+contr_fic)

FIN
Info("Erreur AS/400:" +Milieu(ExtraitChaîne(s1, 7, RC),14)+RC+"Aide :"+RC+
ASErreurAide())
```

Objet de type ASMessage

Message AS/400 d'info ou d'erreur (29 propriétés).

Propriétés de l'objet ASMessage

Nom propriété	Description	Type AS400
messageSeverity	Message severity	BINARY(4)
messageIdentifier	Message identifier	CHAR(7)
messageType	Message type	CHAR(2)
messageKey	Message key	CHAR(4)
messageFileName	Message file name	CHAR(10)
messageFileLibSpec	Message file library specified	CHAR(10)
messageFileLibUsed	Message file library used	CHAR(10)
sendingJob	Sending job	CHAR(10)
sendingJobUserProfile	Sending job's user profile	CHAR(10)
sendingJobNumber	Sending job's number	CHAR(6)
sendingPgmName	Sending program name	CHAR(12)
dateSent	Date sent	CHAR(7)
timeSent	Time sent	CHAR(6)
microSecondes	Microseconds	CHAR(6)
sendingUserProfile	Sending user profile	CHAR(10)
textCCSID	CCSID conversion status indicator for text	BINARY(4)
dataCCSID	CCSID conversion status indicator for data	BINARY(4)
alertOption	Alert option	CHAR(9)
msgOrMsgHelpCCSID	CCSID of message or	BINARY(4)

	message help	
rplcmtDataOrImpromptuMsgTxtCCSID	CCSID of replacement data or impromptu message text	BINARY(4)
RplcmtDataOrImpromptuMSgtxtLenRet	Length of replacement data or impromptu message text returned	BINARY(4)
RplcmtDataOrImpromptuMSgtxtLenAvail	Length of replacement data or impromptu message text available	BINARY(4)
msgReturnedLen	Length of message returned	BINARY(4)
msgAvailableLen	Length of message available	BINARY(4)
msgHlpRetLen	Length of message help returned	BINARY(4)
msgHlpAvailableLen	Length of message help available	BINARY(4)
rplcmtDataOrImpromptuTxt	Replacement data or impromptu text	CHAR(*)
message	Message	CHAR(*)
messageHelp	Message help	CHAR(*)

Objet de type ASMessageHelp

Squelette de message AS/400 avec paramètres positionnels, leurs types et leurs tailles (32 propriétés dont 2 tableaux de paramètres).

Propriétés de l'objet ASMessageHelp

Nom propriété	Description	Type AS400
messageSeverity	Message severity	BINARY(4)
alertIndex	Alert index	BINARY(4)
alertOption	Alert option	CHAR(9)
logIndicator	Log indicator	CHAR(1)
messageID	Message ID	CHAR(7)
nbOfSubstituionVars	Number of substitution variable formats	BINARY(4)
CCSIDconversionStatusIndicText	CCSID conversion status indicator of text	BINARY(4)
CCSIDconversionStatusIndicRplcmtData	CCSID conversion status indicator of replacement data	BINARY(4)
CCSIDTextReturned	CCSID of text returned	BINARY(4)
replyType	Reply type	CHAR(10)
maxReplyLen	Maximum reply length	BINARY(4)
maxReplyDecimalPosition	Maximum reply decimal	BINARY(4)

	positions	
nbValidReplyValues	Number of valid reply values entries returned	BINARY(4)
nbSpecialValidReplyValues	Number of special reply values returned	BINARY(4)
messageCreationDate	Message creation date	CHAR(7)
messageCreationLevelNumber	Message creation level number	BINARY(4)
messageModificationDate	Message modification date	CHAR(7)
messageModificationLevelNumber	Message modification level number	BINARY(4)
storedMessageCCSID	Stored CCSID of message	BINARY(4)
nbDumpListEntries	Number of dump list entries returned	BINARY(4)
defaultPgmName	Default program name	CHAR(10)
defaultPgmLibName	Default program library name	CHAR(10)
defaultReply	Default reply	CHAR(*)
message	Message	CHAR(*)
messageHelp	Message help	CHAR(*)
subVarFormat	Substitution variable formats	CHAR(*)
validReplies	Valid reply value entries	CHAR(*)
specialValidReplies	Special reply value entries	CHAR(*)
lowerRangeReplyValues	Lower range reply value	CHAR(*)
upperRangeReplyValues	Upper range reply value	CHAR(*)
relationalTestEntryFormat	Relational test entry	CHAR(*)
dumpListEntries	Dump list entries	CHAR(*)

Objet de type ASMessageHistoryLogCollection

Ce type d'objet est utilisé comme paramètre de la fonction `ASMessageHistoryLog`. Il s'agit d'une collection d'objets **ASMessageHistoryLog**.

Propriétés de l'objet ASMessageHistoryLog

Nom propriété	Description	Type AS400
msgSeverity	Message Severity	BINARY(4)
msgId	Message ID	CHAR(7)
msgTypeId	Message Type ID (01, 02, 03...)	CHAR(2)
msgType	Message Type as text (e.g. for 01 : Completion, 02: Diagnostic, …)	CHAR(10)

msgFileName	Message File Name	CHAR(10)
msgFileLib	Message File Library	CHAR(10)
dateSent	Date sent	CHAR(7)
timeSent	Time Sent	CHAR(6)
fromJob	From Job	CHAR(10)
fromJobUser	From Job user	CHAR(10)
fromJobNumber	From Job number	CHAR(6)
fromUser	From User	CHAR(10)
dataStatus	Data Status (OK, Unauthorized, Damaged, ...)	CHAR(1)
firstLevelTxt	First level Text message	CHAR(*)

Objet de type ASMessageHistoryLogFilter

Ce type d'objet est utilisé comme paramètre de la fonction `ASMessageHistoryLog`.

Propriétés de l'objet ASMessageHistoryLogFilter

Nom propriété	Description	Type AS400
startDateLog	Start date	CHAR(10)
startTimeLog	Start time (accurate to microsecond)	CHAR(10)
endDateLog	End date	CHAR(10)
endTimeLog	End time (accurate to microsecond)	CHAR(10)
iMsgSeverity	Specific message severity (from 0 to 99)	BINARY(4)
arrMsgIds	Specific message IDs (up to 100 message IDs, e.g. CPF1124)	CHAR(*)
bOmitMsgIds	If (previously defined) message IDs list should be SELECTED or OMITTED	CHAR(*)
arrJob	Specific JOBs, up to 5 JOB filters (could be defined by : jobname, username/jobname or jobnumber/username/jobname)	CHAR(*)
arrMsgTypes	Specific message types list, up to 9 types (9/9) (Message status likes : *COMP Completion, *RPY Reply, ...)	CHAR(*)
bOmitMsgTypes	If (previously defined) message types list should be SELECTED or OMITTED	CHAR(*)

Fonctions Gestion des objets

ASObjectList

Liste les objets d'une librairie AS/400.

Syntaxe

```
Result = ASObjetsList(ASObjetCollection, Librairie, nom, Type, Connexion)
```

Paramètres

Result

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

ASObjectCollection

Variable de type AsObjectCollection qui va contenir les résultats

Librairie

Filtre selon le nom de la librairie, valeurs spéciales *ALL, *ALLUSR, *CURLIB, *LIBL, *USRLIBL. La valeur *LIBL est utilisée si aucune valeur n'est précisée.

Nom

Filtre selon le nom de l'objet, valeurs spéciales *ALL, *ALLUSR, *IBM. La valeur *ALL est utilisée si aucune valeur n'est précisée.

Type

Filtre selon le type de l'objet, valeur spéciale *ALL. Celle-ci est spécifiée si aucune valeur n'est précisée.

Connexion

Connexion - nom de la connexion

Exemple

```
listObjet est un ASObjetCollection

//comportement par défaut liste les objets de la libl.
SI PAS ASObjetsList(listJob,"","", "", MaConnexion1)
 Erreur(ErreurInfo(errComplet))
FIN

// liste les programmes de la bibliothèque CR
SI PAS ASObjetsList(listJob,"CR","", "*PGM", MaConnexion1)
 Erreur(ErreurInfo(errComplet))
FIN
```

Echange de variables

Echange de variables WinDev - introduction

Introduction

Le système d'échanges de variables WinDev est un nouveau système d'échange d'informations avec les programmes AS/400. Ce système est alternatif au classique passage de paramètres.

Les avantages par rapport au passage de paramètres classique sont les suivants :

- Maintenance plus facile : on peut ajouter de nouvelles variables à échanger de part et d'autre sans provoquer de plantage brutal. Le système est contrôlable de chaque côté (gérer l'absence d'une variable par exemple).
- Souplesse de type de données : Gestion des types de données de chaque côté facilité. On peut par exemple avoir une chaîne côté WinDev et un champ de type « pack » côté AS/400.
- Type de données tableau, liste, etc. facilement gérable (ajout élément par élément côte AS/400 par exemple)
- Performance optimale : seules les données modifiées transitent !

L'inconvénient par contre est de devoir lier le programme AS/400 à un programme de service Easycom, afin d'avoir accès aux variables WinDev. Si cela représente un problème, nous suggérons un programme frontal pour séparer la partie utilisant Easycom de la partie native pure.

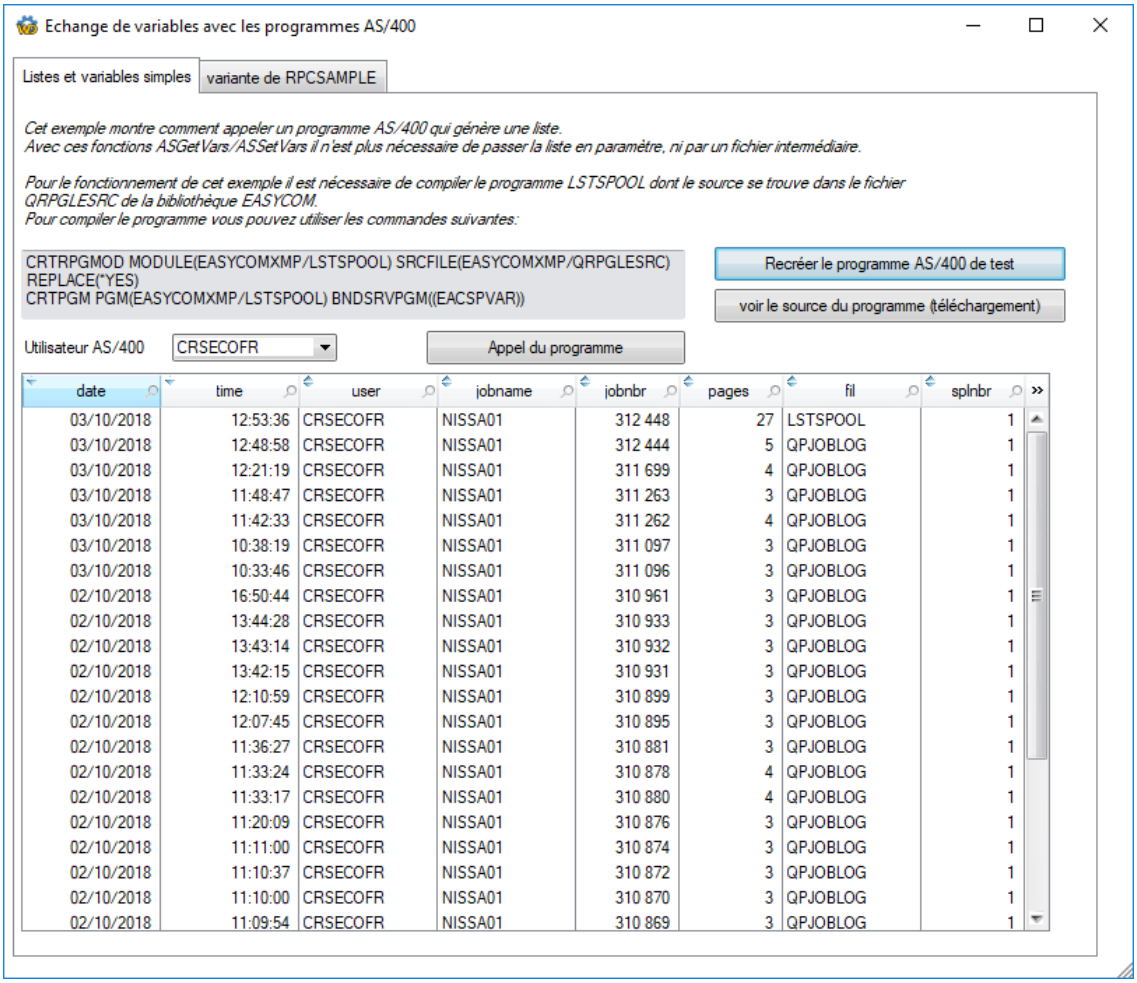
Prérequis

Les pré-requis sont les suivants :

- Partie serveur Easycom (AS/400) version 4.60.32 minimum
- OS/400 V5R3 minimum

Exemples

Un exemple WinDev montre comment utiliser l'échange de variables pour appeler une variante de RPCSAMPLE ainsi qu'un programme retournant une liste de spool.



L'exemple WinDev se nomme 'PgmVars'. Un premier exemple d'appel est en RPG tandis que l'autre appelle un programme CL. Les sources des programmes AS/400 sont dans QCLSRC et QRPGLESRC.

Programmation côté WinDev

Programmation côté WinDev

Le principe est le suivant : avant l'appel de programme, et après celui-ci, on décide quelles variables locales (ou globales) seront accessibles pour le programme AS/400 (RPG, CL, …).

- ASSetVars permet de transmettre une variable au programme. Actuellement cette variable peut être un type simple, une structure, un tableau à une ou deux dimensions de types simples, ou bien un tableau de structures.
- ASGetVars permet de récupérer la valeur d'une variable retransmise directement vers une variable du W-langage.

ASSetVars

La syntaxe de la fonction ASSetVars est la suivante :

```
bResult = ASSetVars(Nom_Variable [, Connexion], Valeur, [Valeur2, …])
```

Paramètres

bResult

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

Nom_Variable

Chaîne – Correspond au nom de variable pour le programme qui va être appelé.

Connexion [optionnel]

Connexion - Nom de la connexion.

Valeur, Valeur2, …

Variable(s) du W-langage ou valeur(s) littérale(s).

La structure des données accessible par les programmes AS/400 reflètera la structure des variables locales du W-langage.

ASGetVars

La syntaxe de la fonction ASGetVars est la suivante :

```
bResult = ASGetVars(Nom_Variable [, Connexion], Valeur, [Valeur2, …])
```

Paramètres

bResult

Booléen - Vrai si la commande a réussi, Faux en cas d'erreur.

Nom_Variable

Chaîne – Correspond au nom de variable modifiée par le programme appelé.

Connexion [optionnel]

Connexion - Nom de la connexion

Valeur, Valeur2, …

Variable(s) du W-langage à modifier.

Remarques :

Les variables du W-langage seront modifiées si leur structure reflète la structure définie par le programme AS/400.

Si une variable a été envoyée par ASSetVars, mais n'a pas été modifiée par le programme, celle-ci demeure accessible, bien que n'ayant pas été retransmise en retour.

Le programme AS/400 peut définir des variables en retour sans qu'elles ait été fixées en entrée par ASSetVars.

Programmation côté AS/400

Programmation côté AS/400

La programmation côté AS/400 s'effectue via l'appel à des fonctions d'un programme de service.

Les spécifications d'appel de ces fonctions se trouvent dans le fichier source QRPGLESRC.

Ces fonctions permettent d'affecter les variables fournies par WinDev à des variables du langage natif, ou bien de préparer les variables qui pourront être lues dans WinDev.

Voici les fonctions actuellement disponibles :

- **variable_get** : récupère une variable WinDev vers une zone mémoire du programme
- **variable_set :** affecte une zone mémoire du programme vers WinDev
- **variable_get_next :** occurrence suivante de la variable WinDev (si tableau)
- **variable_set_next:** occurrence suivante de la variable WinDev (si tableau). Ajoute une occurrence si nécessaire
- **variable_get_data:** comme variable_get mais avec description par variable_init_xxx
- **variable_get_data_next :** comme variable_get_next mais avec description par variable_init_xxx
- **variable_set_data** : comme variable_set mais avec description par variable_init_xxx
- **variable_set_data_next :** comme variable_set_next mais avec description par variable_init_xxx
- **variable_init_pcml** : initialise une description de variable depuis un PCML
- **variable_init_ds** : initialise une description de variable depuis un fichier externe
- **variable_init_desc:** initialise une description de variable avec une description en paramètre
- **variables_config:** configure les options du système d'échanges de variables
- **variables_errinfo :** récupère les informations supplémentaires en cas d'erreur

Description des données natives

La description de données natives s'effectue en utilisant la syntaxe suivante :

Pour un type simple :

```
Nom_var:type[taille[.decimal]]
```

Pour une structure de données :

```
(Nom_var1:type1[taille1[.decimal1]]) (Nom_var2:type2[taille2[.decimal2]])…..
```

Nom_var est le nom de la variable correspondant côté client (WinDev)

Type est égal à une lettre correspondant au type, dans la nomenclature DDS. Les types actuellement reconnus sont les suivants :

- A : chaîne de caractères. La taille est obligatoire.
- H : chaîne de caractères sans conversion. La taille est obligatoire.
- P : décimal condensé. La taille en nombre total de chiffres significatifs est obligatoire. La taille de la partie décimale est facultative.
- S : décimal étendu. La taille en nombre total de chiffres significatifs est obligatoire. La taille de la partie décimale est facultative.
- U : entier non signé. La taille fournie correspond au nombre de chiffres, et induit la taille en octets. Le nombre de chiffres peut être égal à 3, 5, 10, ou 20, ce qui correspond respectivement à 1, 2, 4 ou 8 octets (soit 8, 16, 32 ou 64 bits). La taille par défaut est 5, donc 4 octets.
- I : entier signé. La taille fournie correspond au nombre de chiffres, et induit la taille en octets. Le nombre de chiffres peut être égal à 3, 5, 10, ou 20, ce qui correspond respectivement à 1, 2, 4 ou 8 octets (soit 8, 16, 32 ou 64 bits). La taille par défaut est 5, donc 4 octets.
- B : entier signé. Alternative à 'I'. Le nombre de chiffres est interprété comme suit : s'il est <= à 4 il s'agit d'un entier sur 2 octets (16 bits), autrement il s'agit d'un entier sur 4 octets (32 bits).

- F : type flottant. La taille correspond au nombre d'octets (4 ou 8), et est par défaut égale à 8, soit 64 bits, ce qui correspond au type « double » sur PC.
- D : type date. Date au format *ISO (aaaa-mm-jj).
- T : type time. Heure au format hh.mm.ss
- Z : type timestamp (date et heure), au format *ISO, soit aaaa-mm-jj-hh.mm.ss.MMMMMM.

Exemples :

'dtest3:D' : date, nom de variable 'dtest3'.

'numfact:U20' : entier non signé sur 8 octets, nom de variable 'numfact'.

(ident_a:P9)(code:A10)(date_fin:D) : structure composée de 3 éléments, dont les membres côté PC se nomment 'ident_a', 'code' et 'date_fin', et dont les types locaux sont P9, A10 et date (dans ce cas le nom de la variable WinDev est donné à part).

simple (directe)

variable_get/variable_set/variable_get_next/variable_set_next

Manipule une variable venant de WinDev depuis ou vers une variable locale du programme natif.

variable_get lit la variable vers un pointeur RPG, et variable_set fait le contraire.

Si la variable est de type tableau, variable_get_next permet d'accéder successivement aux occurrences. variable_set_next permet d'ajouter des occurrences.

Remarque : Il faut utiliser variable_get avant variable_get_next, et variable_set avant variable_set_next.

Prototype RPG (ici abrégé en « var_get »):

```
Dvar_get PR 10I 0 ExtProc('variable_get')
D txt_desc * value options(*string)
D data * value
```

txt_desc est une chaîne de caractère de taille indéterminée (terminé par 0), contenant la description de "data" ainsi que le nom de la variable WinDev.

data est un pointeur vers la variable réceptrice.

La fonction retourne un entier différent de zéro en cas d'erreur

Exemple d'utilisation (ici en RPGLE):

```
Dcode_etendu s 6P 0
[…]
/free
iRet = var_get('codeetendu:P6': %addr(code_etendu));
```

Cet exemple lit la variable windev « codeetendu », et l'affecte à la variable locale « code_etendu », de type pack 6.

structurée

Variable_init_desc

Initialise la description d'une variable composée (structure de données).

Prototype RPG (abrégé en var_init_desc) :

```
Dvar_init_desc PR 10I 0 ExtProc('variable_init_desc')
D var_name * value options(*string)
D var_desc * value options(*string)
```

var_name correspond au nom de la variable côté PC.

var_desc correspond à la description des champs de la structure.

Chaque élément de var_desc doit respecter d'une par le nom dans la structure côté PC, et d'autre part le type de données et la taille côté AS/400. Les éléments doivent être dans l'ordre de la DS côté AS/400.

Par contre, certains éléments peuvent ne pas être encore présents dans la structure WinDev. De même que des éléments de la structure WinDev peuvent ne pas figurer dans le programme AS/400. Cela permet dune certaine souplesse d'évolutivité de part et d'autre. L'ordre des champs n'a pas besoin de suivre l'ordre défini dans WinDev (mais doit suivre impérativement l'ordre de la DS AS/400).

Par conséquent, le fait que la définition soit cohérente côté AS/400 suffit pour garantir un bon fonctionnement.

variable_get_data/variable_set_data/variable_get_data_next/ variable_set_data_next

Manipule une variable venant de WinDev depuis ou vers une variable locale du programme natif.

Ces fonctions sont prévues pour manipuler des structures de données.

Un appel préalable à variable_init_desc, variable_init_ds ou variable_init_pcml est nécessaire.

variable_get_data lit la variable vers un pointeur RPG, et variable_set_data fait le contraire.

Si la variable est de type tableau, variable_get_data_next permet d'accéder successivement aux occurrences. variable_set_data _next permet d'ajouter des occurrences.

Remarque : Il faut utiliser variable_get_data avant variable_get_data _next, et variable_set_data avant variable_set_data _next.

Prototype RPG :

```
Dvar_get_data PR 10I 0 ExtProc('variable_get_data')
D var_name * value options(*string)
D data * value
```

`var_name` correspond au nom de la variable côté PC.

`data` est un pointeur vers la variable réceptrice, qui peut être un élément de tableau de structure, ou une structure simple.

La fonction retourne un entier différent de zéro en cas d'erreur

Exemple d'utilisation (ici en RPGLE):

```
Dpartenaire DS
D identif 9P 0
D cdealpha 10
D raisonsoc 30
D absorbes 70
[…]
/free
// define array of structure result
iRet = var_init_desc('wpart':
'(identifiantPartenaire:P9)' +
'(codealphaPartenaire:A10)' +
'(raisonSociale:A30)' +
'(absorbes:A70)'
);
iRet = var_set_data('wpart': %addr(partenaire));
```

Cet exemple lit la variable windev 'wpart', et l'affecte à la DS locale 'partenaire'

variable_init_ds

Initialise la description d'une variable composée à l'aide d'une DS externe.

Protype RPG (abrégé en var_init_ds) :

```
Dvar_init_ds PR 10I 0 ExtProc('variable_init_ds')
D var_name * value options(*string)
D ds_name * value options(*string)
```

`var_name` correspond au nom de la variable côté PC.

`ds_name` correspond au nom du fichier (physique ou logique), dont la description correspond à la DS que l'on veut affecter.

Remarques :

- Seuls les champs du fichier externe dont les noms sont identiques côté programme WinDev auront une correspondance. Les non-correspondant seront initialisés à blanc, en cas de 'get'.
- Entre le PC est le fichier externe, seuls les noms doivent correspondre ; l'ordre n'a pas d'importance.
- La DS et le fichier externe doivent bien entendu correspondre

variable_init_pcml

Fonctionne comme variable_init_desc mais avec un fichier PCML en entrée.

Protype RPG (abrégé en var_init_pcml) :

```
Dvar_init_pcml PR 10I 0 ExtProc('variable_init_pcml')
D var_name * value options(*string)
D pcml_name * value options(*string)
D struct_name * value options(*string)
```

`var_name` correspond au nom de la variable côté PC.

`pcml_name` correspond au nom du fichier IFS contenant le code PCML

`struct_name` correspond au nom de la structure à utiliser dans le fichier PCML

Fonctions additionnelles

variables_config

Modifie les options locales au programme natif concernant l'échange de variables.

Protype RPG (abrégé en var_config) :

```
Dvar_config PR 10I 0 ExtProc('variables_config')
D cfg_string * value options(*string)
```

`cfg_string` correspond à une chaîne de caractère correspondant aux options à modifier

La syntaxe de cfg_string est la suivante :

Cfgopt1=valeur1;cfgopt2=valeur2 ; …

Seule une option est disponible :

- Msg détermine si un message est généré en cas d'erreur. Valeurs possibles : 0 ou 1. Valeur par défaut : 1.
    - Si la valeur est 1, un message de type « *ESCAPE » est généré en cas d'erreur, et doit être rattrapé par un mécanisme de traitement des exceptions (soit, en RPG un monitor/on-error, par exemple). S'il n'y a pas de mécanisme de rattrapage d'exception, le travail Easycom passe en état MSGW jusqu'à ce que l'on réponde au message par annulation ou ignorer.
    - Remarque : si msg=0, un message de type *INFO est tout de même envoyé.

Exemple :

```
/free
iRet = var_config ('msg=0');
```

Variables_errinfo

Permet de lire les informations sur la dernière erreur qui a été renvoyée par les fonctions variables_xxx.

Protype RPG (abrégé en var_errinfo) :

```
Dvar_errinfo PR 10I 0 ExtProc('variables_errinfo')
D errmsg 1 options(*varsize)
D errmsglen 10I 0 value
```

`errmsg` est le message d'erreur reçu

`errmsglen` est la taille de réception pour le message d'erreur

Cette fonction peut être appelée en cas d'interception d'exception, ou bien en test de retour des fonctions.

Exemple :

```
monitor;
[…]
on-error 00202;
// retreive error message
var_errinfo(msg: 30, 30);
```

```
// display it...
dsply msg;
endmon;
```

Exemples

Exemples - Introduction

Un projet Exemples Easycom vous est proposé et illustre l'essentiel des opérations et des fonctions de l'accès natif AS/400.

Gestion des connexions
Appel de programme
Utilisation des DataQueues
Utilisation de DataArea
ASAppelRTV et ASRésultatRTV
Requêtes et SQL
Zone mémoire, filtres et vues
Transactions
Gestion d'erreur

Gestion des connexions

Cet exemple présente les fonctions d'ouverture et de fermeture de connexions, avec ou sans paramétres, avec des options complémentaires par les infos étendues (JOBNAME), la fonction ASUtilisateur pour donner les droits d'un autre profil au job.

Commentaires de l'exemple :

- L'ouverture de la connexion définie dans l'analyse est automatiquement proposée à l'ouvertures des fichiers qui lui sont associés. La fermeture du dernier fichier va également fermer la connexion.
- Appel de `HOuvreConnexion` sans paramètres : affichage de la boite de dialogue Easycom.
- La fonction `ASUtilisateur` donne au job les droits d'un profil connu uniquement par le programme. Elle ne change pas la propriété USER mais CURUSER.
- Appel de `HOuvreConnexion` avec des champs personnalisés.
- Adresse IP ou nom peut être vide si l'adresse a été paramétrée par Easycom Configuration.
- On peut aussi modifier dynamiquement certaines propriétés de la connexion avec les infos étendues.
- La fonction `HFermeConnexion` va arrêter le job et fermer la connexion. Les fichiers sont évidemment fermés aussi.

Extraits du code

```
SI bconnecté ALORS HFermeConnexion(PrinciConnexion)
PrinciConnexion..InfosEtendues="<EASYCOM>"+RC+"JOBNAME=+SAI_JOB+RC+"<EASYCOM>"
SI HOuvreConnexion(PrinciConnexion) ALORS
 bconnecté=Vrai
 Message("Connecté...")
SINON
 bconnecté=Faux
 Message("Non connecté...")
FIN
```

Appel de programmes

Cet exemple simple est constitué de 5 champs avec liaison sur le fichier RPCSAMPLE qui a été décrit avec le Constructeur RPC/DTAQ et importé dans l'analyse.

Ce fichier correspond à un programme qui attends 5 paramètres en entrée et qui retourne 3 paramètres en sortie.

```
EcranVersFichier("",RPCSAMPLE)
ASLanceRPC(RPCSAMPLE)
FichierVersEcran("",RPCSAMPLE)
```

La connexion utilisée est la connexion associée au fichier dans l'analyse.

On peut également utiliser `HAjoute`.

Il utilise un programme CL très rudimentaire, RPCSAMPLE qui se trouve dans la bibliothèque Easycom et dont le source se trouve dans QCLSRC de la même bibliothèque :

```
0001.00              PGM        PARM(&VAL1I &STR1I &VAL2IO &STR2IO &VAL3O)
0002.00
0003.00              DCL        VAR(&VAL1I) TYPE(*DEC) LEN(5 2)
0004.00              DCL        VAR(&VAL2IO) TYPE(*DEC) LEN(5 2)
0005.00              DCL        VAR(&VAL3O) TYPE(*DEC) LEN(10 4)
0006.00              DCL        VAR(&STR1I) TYPE(*CHAR) LEN(20)
0007.00              DCL        VAR(&STR2IO) TYPE(*CHAR) LEN(30)
0008.00
0009.00              CHGVAR     VAR(&VAL3O) VALUE(&VAL1I * &VAL2IO)
0010.00              CHGVAR     VAR(&VAL2IO) VALUE(&VAL1I + &VAL2IO)
0011.00              CHGVAR     VAR(&STR2IO) VALUE(&STR1I)
```

Ce programme attend 5 variables en entrée, 2 chaînes et 3 numériques.

Le texte de la première chaîne est copié dans la seconde.

La valeur du deuxième opérateur est additionnée à celle du premier.

Le troisième opérateur (en sortie seule) est le résultat de la multiplication du premier par le second.

Les variables se retrouvent dans la description du fichier avec leur type et leur nature (I, IO ou O)

Utilisation des DataQueues

Cet exemple illustre l'écriture et la lecture (directe ou temporisée) dans deux dataqueues (FIFO et KEYED) décrites depuis l'utilitaire "Constructeur RPC-DTAQ".

Ecriture :

```
HAjoute(DTAQ_FIFO)
```

Lecture FIFO :

```
HLitRecherchePremier(DTAQ_FIFO,Timeout,valeurTimeout)
```

Lecture KEYED :

```
HLitRecherchePremier(DTAQ_KEY,DTAQ_KEY,cléco)
```

Extrait du code

```
// lecture temporisée sur la valeur en secondes de timeout

Cléco, Timeout, filer, order, key, sData sont des chaînes
timeout=Val(SAI_TIMEOUTK)
filer=""
order=Gauche(COMBO_OP[COMBO_OP],2)+"" // opérateur EQ, GT, GE, LT ou LE
key=SAI_CLE
cléco=HConstruitValClé(DTAQ_KEY,DTAQ_KEY,timeout, filer, order, key)
HLitRecherche(DTAQ_KEY,DTAQ_KEY,cléco)
SI HTrouve ALORS
 SAI_MSG2=DTAQ_KEY.Data
SINON
 Info("Pas trouvé")
FIN
```

Utilisation de DataArea

Cet exemple utilise un programme CL de la bibliothèque Easycom pour lire dans un Data Area de la bibliothèque Easycom et la fonction ASExec pour y écrire.

Il est également possible de lire une DTAARA avec les fonctions ASAppelRTV et ASRésultatRTV.

Ecriture par ASExec

```
sMsg est une chaîne
sMsg="Ce message a été décrit dans la Data Area depuis votre programme WinDev !"
ASExec("CHGDTAARA DTAARA(EASYCOM/DTAARA (1 200)) VALUE('"+sMsg+"') ")
```

Lecture par le programme RTVDTAARA

```
SI INT_LDA[1] ALORS
 RTVDTAARA.Dtaara="*LDA"
SINON
 RTVDTAARA.Lib=SAI_LIB
 RTVDTAARA.Dtaara=SAI_DTA
 SI (SAI_LIB="" OU SAI_DTA="") ALORS
  Info("Bibliothèque et/ou nom de la data area vides...")
  RETOUR
 FIN
FIN

SI SAI_TAILLE=0 ALORS
 RTVDTAARA.Size=200
SINON
 RTVDTAARA.Size=SAI_TAILLE
FIN

ASLanceRPC(RTVDTAARA)
SAI_VAL=RTVDTAARA.Value
FichierVersEcran()
```

Lecture par un ASRésultatRTV

```
// Lecture du résultat
scmd= " RTVDTAARA DTAARA(EASYCOM/DTAARA) RTNVAR(&VAR1)"

bresult = ASAppelRtv(scmd)
Resultat = ASResultatRtv("RC")

SI Resultat = "0" ALORS
 var1=ASResultatRtv("VAR1")
FIN
```

ASAppelRTV et ASRésultatRTV

Utilisation des fonctions ASAppelRTV et ASRésultatRTV pour récupérer les propriétés du job ou du membre d'un fichier.

RTVJOBA

```
LigneCmd est une chaîne
LigneCmd="CCSID=DEC(5 0);RTVJOBA JOB(&JOB) USER(&USER) USRLIBL(&USRLIB) SYSLIBL(&SYSLIB)
CCSID(&CCSID) CURLIB(&CURL)"

// les variables décimales doivent être déclarées de même pour un char de taille
indéterminée

SI PAS ASAppelRtv(LigneCmd)ALORS
```

```
 Info(ErreurInfo(errRésumé))
FIN

Info("Commande envoyée avec succès."+RC+"Ce retour ne signifie cependant pas la réussite de l'opération elle-même..."+RC+"Il faut tester la variable RC.")

sResultat est une chaîne
sResultat = ASResultatRtv("RC")
SI sResultat = "0" ALORS
 SAI_USR=ASResultatRtv("USER")
 SAI_USRLIBL=ASResultatRtv("USRLIB")
 SAI_SYSLIBL=ASResultatRtv("SYSLIB")
 SAI_JOB=ASResultatRtv("JOB")
 SAI_CCS=ASResultatRtv("CCSID")
SINON
 Info("Erreur de la commande : "+sResultat)
FIN
```

RTVMBRD

```
cmd est une chaîne
sAsdate est une chaîne
Wddate est une Date

cmd="NBREG=DEC(10 0);RTVMBRD FILE("+SAI_biblio+"/"+SAI_file+") CRTDATE(&CRDATE) NBRCURRCD(&NBREG) FILETYPE(&TYPE)"
ASAppelRtv(cmd)
sResultat est une chaîne
sResultat = ASResultatRtv("RC")
SI sResultat <> "0" ALORS
 SI sResultat="CPF9812" ALORS Info("Fichier non trouvé") SINON Info("Erreur de la commande : "+sResultat)
SINON
 // récupérer et formater la date
 sAsdate=ASResultatRtv("crdate")
 sAsdate=Gauche(sAsdate,7)
 SI Gauche(sAsdate,1)="0" ALORS
  wddate="19"+Droite(sAsdate,6)
 SINON
  wddate="20"+Droite(sAsdate,6)
 FIN
 SAI_DATE=DateVersChaîne(wddate,"JJJJ JJ MMMM AAAA")
 SAI_NBR=ASResultatRtv("NBREG")
 SAI_TYP=ASResultatRtv("TYPE")

FIN
```

Requêtes et SQL

Trois onglets proposent les trois types de requêtes :

- directe (chaîne SQL libre)
- préparée (initialisation dans un premier temps, exécution(s) dans un second temps)
- pré-définie (requête créée et nommée dans l'éditeur de requêtes)

Il est possible d'utiliser toutes les fonctionnalités et mots clés SQL.

Directe

```
Marequete est une Source de Données
srequete est une chaîne
SI HExécuteRequêteSQL(Marequete,hRequêteDéfaut,"SELECT * FROM SP_CUST") ALORS
 Info(HListeRubrique(Marequete))
 sResultat est une chaîne
```

```
 HLitPremier(Marequete)
 TANTQUE PAS HEnDehors(Marequete)
  sResultat=sResultat+Marequete.cust_id+Marequete.firstname+Marequete.lastname+RC
  HLitSuivant(Marequete)
 FIN
 Info(sResultat)
FIN
```

Préparée

```
Marequete est une Source de Données
srequete est une chaîne

// préparation
srequete="UPDATE Sp_cust SET City = :Param1 WHERE Sp_cust.City =
SI HPrépareRequêteSQL(Marequete,PrinciConnexion,hRequêteSansCorrection,srequete)
ALORS
 Info("Requête préparée...")
SINON
 Info(HErreurInfo())
FIN


// Exécution
Marequete.Param1=SAI_PARAM1
Marequete.Param2=SAI_PARAM2
SI SAI_PARAM1="" OU SAI_PARAM2="" ALORS
 Info("Il faut renseigner les paramètres...")
 RETOUR
FIN
// Initialisation de la requête "Client"
SI PAS HExécuteRequêteSQL(Marequete)
 Info(HErreurInfo(hErrNatif))
FIN
```

Pré-définies

La requête REQ_Exemple2 a été créée dans l'éditeur de requêtes.
Elle prend deux paramètres qu'on initialise avant l'appel par HExecuteRequete.

Attention !!! Les quotes ne sont pas automatiquement insérées et doivent être ajoutées avant et après les valeurs :

```
REQ_Exemple2.Param1="'"+SAI_P1+"'"
REQ_Exemple2.Param2="'"+SAI_P2+"'"

SI HExécuteRequête(REQ_Exemple2,PrinciConnexion,hRequêteSansCorrection) ALORS
 HLitPremier(REQ_Exemple2,Cust_id)
 Info("Requête exécutée..."+RC+REQ_Exemple2.Cust_id+" - "+REQ_Exemple2.Order_id+" - "+
REQ_Exemple2.Order_date)
SINON
 Info(HErreurInfo())
FIN
```

Zone mémoire, filtres et vues

Cet exemple présente une utilisation basique de filtres, sur clé et sur condition et du HCréêVue et illustre une manière de charger des données statiques d'une table dans une zone mémoire afin de limiter les accès serveur dans la suite du programme.

```
// zone mémoire
liste_villes est une Source de Données
sRequete est une chaîne
sRequete="SELECT DISTINCT CITY FROM EASYCOM/SP_CUST"
HExécuteRequêteSQL(liste_villes, PrinciConnexion,hRequêteSansCorrection,sRequete)
```

```
MemCrée("LST_VILLES")
HLitPremier(liste_villes,"CITY")
TANTQUE PAS HEnDehors(liste_villes)
 MemAjoute("LST_VILLES",liste_villes.City,liste_villes.City)
 HLitSuivant(liste_villes,"CITY")
FIN

// remplir la combo avec la zone mémoire
MemPremier("LST_VILLES")
TANTQUE PAS MemEnDehors("LST_VILLES")
 ListeAjoute(COMBO_LST_VIL,MemSuivant("LST_VILLES"))
FIN
```

Transactions

Pour gérer les transactions les fichiers doivent être journalisés et contenir l'option JOURNALED=TRUE dans les infos étendues.

Cet exemple commence par créer un récepteur et un journal dans la bibliothèque EASYCOM :

```
sCmd est une chaîne
sCmd="CRTJRNRCV JRNRCV(EASYCOM/TMPRCV)"
SI PAS ASExec(sCmd) ALORS
 SI ExtraitChaîne(ErreurInfo(),2,CR)="CPF7010" ALORS Info("Récepteur déjà présent") SINON
Info(ErreurInfo)
FIN

sCmd="CRTJRN JRN(EASYCOM/TMPJRN) JRNRCV(EASYCOM/TMPRCV)"
SI PAS ASExec(sCmd) ALORS
 SI ExtraitChaîne(ErreurInfo(),2,CR)="CPF7015" ALORS Info("Le récepteur contient déjà ce poste
de journal") SINON Info(ErreurInfo)
FIN

sCmd="STRJRNPF FILE(EASYCOM/SP_CUST) JRN(EASYCOM/TMPJRN)"
SI PAS ASExec(sCmd) ALORS
 SI ExtraitChaîne(ErreurInfo(),2,CR)="CPF7030" ALORS Info("Fichier déjà journalisé") SINON
Info(ErreurInfo)
FIN
```

Un parcours simplifié du fichier propose de faire des modifications dans ce dernier, avec ou sans transaction.

Début de transaction

```
SI SQLTransaction(sqlDébut,PrinciConnexion)
 Info("Transaction démarrée")
 btransaction=Vrai
FIN
```

Fin de transaction (COMMIT)

```
SI SQLTransaction(sqlFin,PrinciConnexion)
 Info("Transaction validée")
 btransaction=Faux
SINON
 Info(ErreurInfo())
FIN
```

Fin de transaction (ROLLBACK)

```
SI SQLTransaction(sqlAnnule,PrinciConnexion)
 Info("Transaction annulée")
 btransaction=Faux
SINON
 Info(ErreurInfo())
FIN
```

Gestion d'erreur

Cet exemple provoque différents types d'erreur pour illustrer la récupération des informations importantes (Catégorie, Code, CPF, message court, détaillé...).

Voir Gestion des erreurs.

Erreurs Hyper File

```
serreur est une chaîne
serreur=HErreurInfo(hErrMessageNatif)

SAI_CAT=ExtraitChaîne(serreur,1,TAB)
SAI_CODE=ExtraitChaîne(serreur,2,TAB)
SAI_CPF=ExtraitChaîne(serreur,3,TAB)
SAI_MSG=ExtraitChaîne(serreur,4,TAB)

SAI_AIDE=ASErreurAide()
```

Erreurs sur les fonctions

```
serreur est une chaîne
serreur=ErreurInfo()

SAI_CAT=ExtraitChaîne(serreur,3,RC)
SAI_CODE=ExtraitChaîne(serreur,5,RC)
SAI_CPF=ExtraitChaîne(serreur,2,RC)
SAI_MSG=ExtraitChaîne(serreur,1,RC)

SAI_AIDE=ASErreurAide()
```

Gestion des erreurs et débogage

Gestion d'erreur

La gestion d'erreur est en grande partie intégrée dans les fonctions WinDev.

Les variables de gestion (hErreurBlocage, hErreurIntégrité, hErreurDoublon) sont automatiquement mises à jour.

Note : pour une erreur de doublon, le nom de la rubrique se récupère par HErreurInfo(*hErrRubrique*)

WRKACTJOB

En consultant les propriétés du JOB vous pouvez également obtenir un certains nombres d'informations utiles :

- historique des messages,
- fichiers ouverts,
- verrouillages...

Traces

Les fichiers de trace conservent le détail des opérations effectuées, côté PC ou AS et peuvent être très utiles pour identifier un problème.

Fonctions Easycom

La fonction ASRésultatRTV a une variable spécifique "RC", si la syntaxe de la commande est correcte mais que l'exécution échoue cette variable contiendra le code erreur CPF, sinon elle vaut "0". Il faut donc tester le retour de la fonction (syntaxe et passage correcte) et la valeur de la variable RC.

Les autres fonctions Easycom retournent un booléen, Vrai en cas de succès, Faux en cas d'échec.

A l'exception de ASPropriete, la connexion doit être ouverte avant l'appel de la fonction.

Erreurs de type CPF

Voir les fonctions ASErreurAide et ASErreurDonnee qui remontent le détail d'une erreur "signal", catégorie 2.

ErreurInfo et HErreurInfo

hErreur retourne le numéro de la dernière erreur WinDev rencontrée.

Le code **73001** indique une erreur de l'accès natif. Il faut alors utiliser ErreurInfo ou hErreurInfo pour obtenir le détail de l'erreur.

Les fonctions ErreurInfo et HErreurInfo doivent être appelées immédiatement après la fonction à contrôler.

ErreurInfo (*hErrNatif*) retourne le numéro de l'erreur.

Pour les fonctions Easycom (ASExec, ASLanceRPC....), la fonction ErreurInfo() retourne une chaîne séparée par des CRLF (sauts de ligne) composée de :

- texte complet du message d'erreur,
- texte court du message d'erreur,
- catégorie,
- code principal selon la catégorie,
- code secondaire.

Pour les opérations sur les fichiers ou les requêtes SQL, utiliser HErreurInfo(*hErrMessageNatif*) qui retourne une chaîne séparée par des TAB (tabulations) composée de :

- catégorie,
- code principal selon la catégorie,
- texte court du message d'erreur (code CPF ou SQLSTATE)
- texte long du message d'erreur.

Erreurs fatales

L'appel d'une fonction Easycom (AsExec...) renvoie une erreur fatale si la connexion n'est pas ouverte

Il faut donc gérer les exceptions.

Exemple

```
// on provoque une erreur
ASExec("ADDLIBLE EASYCOM")
ASExec("ADDLIBLE EASYCOM ")
```

ErreurInfo(*errInfo*) renvoie 'CPF2103<RC>2<RC>1<RC>11'

ErreurInfo(*errMessage*) renvoie 'La bibliothèque Easycom existe déjà dans la liste des bibliothèques.<RC>CPF2103<RC>2<RC>1<RC>11'

ErreurInfo(*errResumé*) renvoie 'La bibliothèque Easycom existe déjà dans la liste des bibliothèques'.

La première ligne contient le code AS/400 'Texte'. Nous obtiendrons par exemple : 'CPF2103', si la catégorie est égal à 2.

La deuxième ligne contient la catégorie d'erreur sous forme numérique.

La ligne 3 contient le code 'Natif' Easycom (voir codes erreur Easycom)

La ligne 4 contient le code AS/400 numérique s'il existe.

Vous pouvez obtenir le message clair (pour affichage par exemple) sans les codes en demandant `ErreurInfo(errRésumé)`.

Comment diagnostiquer les erreurs

En cas de problème de connexion, plusieurs éléments peuvent contenir une information sur le problème:

- Dans l'historique du travail EasycomD. Pour l'examiner, utiliser WRKACTJOB, ensuite l'option 5 sur EASYCOMD, puis option 10 (historique du travail), et appuyer F10. Appuyer ensuite sur F1 sur les messages qui paraissent anormaux.
- Dans les messages de EACMSGQ. Pour les visualiser, utiliser DSPMSG EASYCOM/EACMSGQ.
- Dans les messages de QSYSOPR. Pour les visualiser, utiliser DSPMSG QSYSOPR. Les défaillances inattendues ou bien les erreurs de licence apparaissent notamment dans cette file de messages.
- Dans l'OutQ QEZJOBLOG. Un fichier spool est créé dans cette OutQ si le travail Easycom n'a pas réussi à démarrer correctement, ou s'il s'est mal arrêté.

  Pour visualiser le fichier spool, utiliser les commandes suivantes :

  - WRKOUTQ OUTQ(QEZJOBLOG)
  - Appuyer sur F18 (pour aller en fin de liste), et ensuite sur F11.
  - Doit figurer une entrée avec le nom de station ainsi que l'utilisateur correspondant. On peut également s'aider de la date et heure de génération.

  Choisir « 5 » pour visualiser le spool de messages (contient des messages d'information, d'avertissement et/ou d'erreurs).
- Dans le fichier LOGFILE membre LOGFILE de la bibliothèque EASYCOM. Ce fichier contient toutes les défaillances TCP/IP avec les codes d'erreurs correspondants. Pour visualiser le fichier, utiliser : DSPPFM EASYCOM/LOGFILE MBR(LOGFILE)

  Ce fichier peut être téléchargé par FTP ou l'outil Easycom configuration depuis Windows.

En cas d'erreurs pendant le fonctionnement, un fichier log Easycom peut être utile à diagnostiquer le problème. Il peut être mis en place à l'aide de la commande CFGEAC ou bien de l'outil Easycom configuration. Le contenu de ce fichier peut aider à comprendre les raisons de certaines erreurs : visualisation des paramètres, messages d'erreur supplémentaires, …

L'historique du travail Easycom peut également aider. Pour le visualiser, utiliser l'option 5 avec WRKACTJOB, puis option 10 et appuyer sur F10.

Si le travail Easycom s'arrête trop rapidement pour pouvoir visualiser l'historique, utiliser l'option JOBLOG(*YES) de la commande CFGEAC, qui permet d'avoir systématiquement un spool généré dans QEZJOBLOG (voir ci-dessus pour comment le consulter ensuite).

Si le travail EASYCOM ou EASYCOMD tourne sans réponse (état RUN permanent), il faut essayer de visualiser la pille d'appels. Celle-ci est disponible par WRKACTJOB, option 5 sur le travail puis option 11.

En cas d'erreurs de licence (clé), faire DSPSMG QSYSOPR, si les informations affichées sur les client ne sont pas suffisantes.

Fichiers de trace

Le fichier de trace est un fichier "log" qui consigne l'ensemble des opérations internes effectuées, soit côté client par la dll easycom, soit côté serveur par la partie AS400 d'Easycom.

L'activation de la trace réduit significativement les performances et peut créer des fichiers très volumineux, il faut donc restreindre son usage à des analyses ponctuelles.

Pour activer la trace PC il faut renommer la dll principale, pour activer la trace AS, utiliser Easycom Configuration, pour définir le chemin, le niveau de détail, des options (clock) ainsi que la récupération du fichier de trace sur le PC.

Il est également possible de créer une trace pour l'émulateur 5250, pour cela il faut éditer le fichier easycom.ini et rajouter dans la section [EM5250] le nom du fichier :

```
logfile=c:\tracepc5250.txt
```

Voir : activer les fichiers de trace.

Activer le fichier de trace

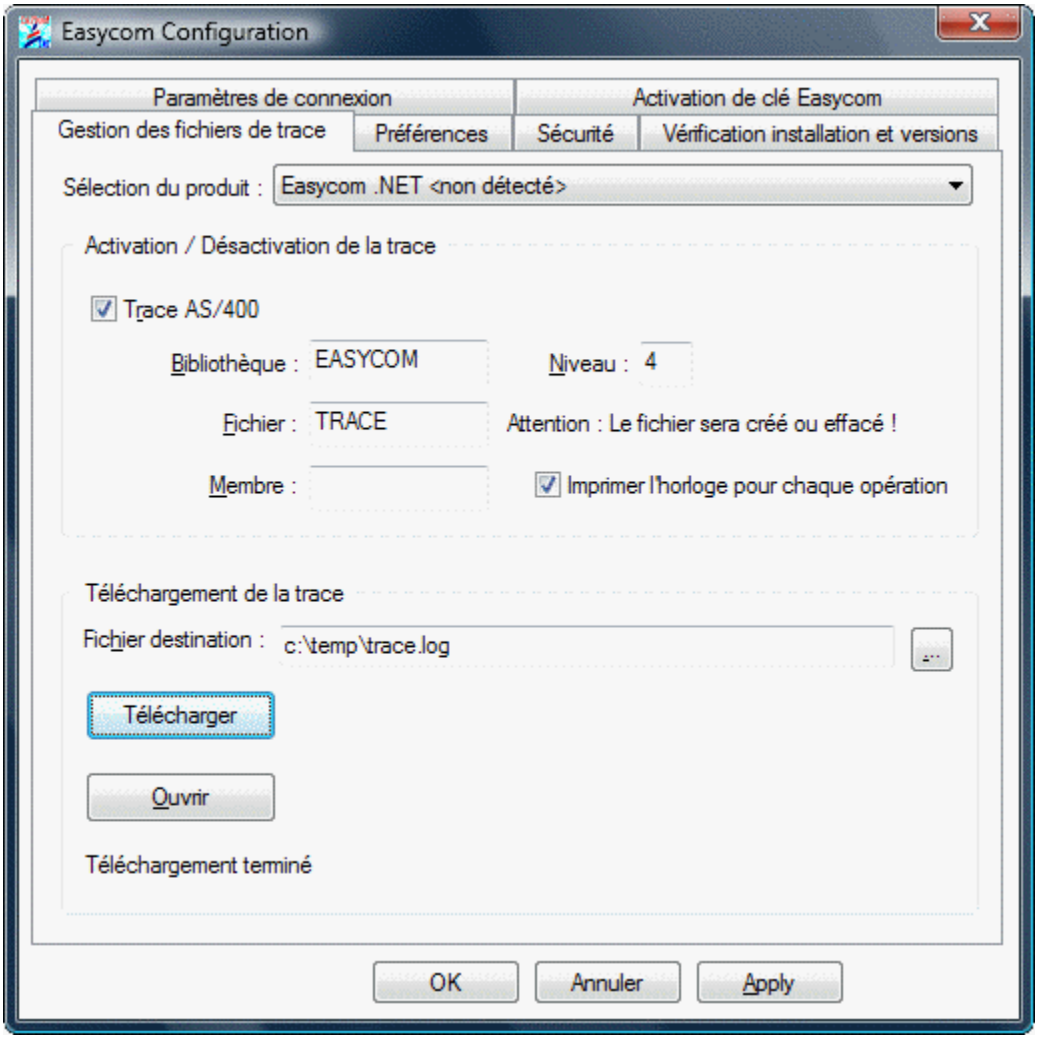
Pour activer la trace AS400

Utiliser l'utilitaire EASYCOM Configuration , onglet Gestion des fichiers de trace.

Vous pouvez définir la bibliothèque et le nom du fichier trace à créer et son niveau de détail (de 1 à 9, le niveau 1 étant le plus sommaire et le niveau 9 le plus détaillé).

L'option "Imprimer l'horloge" permet d'avoir un timestamp devant chaque ligne d'opération.

A partir du même écran vous pouvez récupérer cette trace sur le PC sous la forme d'un fichier texte.



Il est également possible d'activer la trace depuis un terminal par la commande CFGEAC.

Pour activer la trace PC

Procédure d'exception pour identifier des problèmes inhabituels, la trace PC va consigner le détail des opérations effectuées par la dll EASYCOM (eac1100as.dll) dans un fichier texte.

Il faut utiliser une dll particulière (eac1100tr.dll) en la renommant en eac1100as.dll, ces dlls se trouvent par défaut dans le répertoire C:\WinDev 11\Programmes mais peuvent également être présentes dans le répertoire d'une application déployée.

Pour activer la trace PC

- Fermer WinDev (ou WebDev)
- Renommer eac1100as.dll en eac1100as_sauve.dll (par exemple)
- Renommer eac1100tr.dll en eac1100as.dll
- Relancer WinDev (ou WebDev)

Pour désactiver la trace et revenir à un fonctionnement normal :

- Fermer WinDev (ou WebDev)
- Renommer eac1100as.dll en eac1100tr.dll
- Renommer eac1100as_sauve.dll en eac1100as.dll
- Relancer WinDev (ou WebDev)

Pour spécifier le nom du fichier il faut éditer le fichier easycom.ini et compléter la section [WINDEV]

```
[WINDEV]
Trace_PC=C:\TRACE.TXT
```

Pour activer la trace de l'émulateur 5250

Exceptionnellement si vous utilisez le composant 5250 et rencontrez des problèmes vous pouvez créer une trace en éditant le fichier easycom.ini pour compléter la section [EM5250] avec le nom du fichier à créer :

```
[EM5250]
logfile=c:\tracepc5250.txt
```

Fichier de Trace AS/400

Les fichiers de trace permettent de visualiser les opérations réalisées par EASYCOM, côté client ou côté serveur. Le serveur EASYCOM sur l'AS/400 traite des requêtes élémentaires sur les tables ou procédures.

Il reçoit une demande de traitement sur le réseau (Requête), et retourne une réponse.

Les requêtes et réponses peuvent être enregistrées dans un fichier de trace, et ce fichier peut servir à analyser le flux de données qui circulent entre le client et le serveur.

Les lignes commençant par << marquent la réception d'une requête.

Les lignes commençant par >> marquent l'envoie par l'AS/400 de la réponse.

```
<<EACopen(EASYCOM/SP_CUST,4194309,-1) ß Requête.
9 Fields, 0 key fields
EAC_NO_CVT - mode=
>>Ret=1; Err=0; Msg=; Int=0 ß Réponse.
```

Pour la trace AS/400, si l'option d'horodatage a été choisie, toute les requêtes et réponses sont précédées de l'heure sous la forme : hh :mm :ss :mil.

```
<<15:48:45.566: EACread(1,p(2275),91,34144281,(null),0,p(100))
```

Sur la réponse, l'information « Clk=x » indique le temps CPU consommé par l'AS/400 pour le traitement de la requête.

```
>>15:48:45.574: Clk=8, Len=619; Ret=5; Err=0; Msg=; Int=0
```

Début du fichier de trace, commun à toutes les sessions

Cette partie du fichier de trace est toujours la même pour toutes les sessions de EASYCOM.

```
Time is : 03/27/2000 - 17:22:10
Easycom Server Version is : 4.5712, Link is TCP/IP
Client licence is : D$WINDEV55 , Easycom Library is : EASYCOM
JobName=ALBATROS, User=QPGMR , QCCSID=297 Heart Beat freq :10
Easycom Log File TRACE/SR, level 1
-----------------------------------------------------
>>Ret=1; Err=0; Msg=; Int=0
<<RTV_AS_VER(p(4))
>>Ret=4; Err=0; Msg=; Int=0
<<WriteTableEBCDI(49 42 4D 43 43 53 49 44 20 30 20 31 32 35 32 00 00
00 00 00 ...(256))
Build Table from CCSID:0 to 1252
open IBMCCSID01252, IBMCCSID000000000100
<<WriteTableASCII(49 42 4D 43 43 53 49 44 20 31 32 35 32 20 30 00 00
00 00 00 ...(256))
Build Table from CCSID:1252 to 0
open IBMCCSID00000, IBMCCSID012520000100
<<ReadTableASCII(p(256))
>>Ret=0; Err=0; Msg=; Int=
<<ReadTableEBCDIC(p(256))
>>Ret=0; Err=0; Msg=; Int=0

<<EACSqlDeclare(2A 45 41 43 20 43 56 54 20 4E 4F 00 ,12)
Statement:*EAC CVT NO
>>Ret=1; Err=0; Msg=; Int=0
<<EACSqlBegin()
>>Ret=0; Err=0; Msg=; Int=0
```

Trace d'une ouverture d'un fichier Physique

```
<<EACopen(EASYCOM/SP_CUST,4194309,-1)
9 Fields, 0 key fields
EAC_NO_CVT - mode=rr+
>>Ret=1; Err=0; Msg=; Int=0
<<EACgetdesc(1,p(65000),65000,939786240,(null))
>>Ret=9; Err=0; Msg=; Int=0
```

Trace d'une ouverture d'un fichier Logique

```
<<EACopen(EASYCOM/SP_CUST_UN,4194309,-1)
LF with 1 Data Members, 1 Record Formats
9 Fields, 1 key fields
EAC_NO_CVT - mode=rr+
>>Ret=2; Err=0; Msg=; Int=0
<<EACgetdesc(2,p(65000),65000,939786240,(null))
>>Ret=9; Err=0; Msg=; Int=0
```

Trace d'une lecture d'enregistrements de fichier

```
 <<EACread(2,p(816),102,34144264,(null),0,p(32))
VERB=_EAC_NEXT LOCK=OFF RECS=8 FILE=EASYCOM/SP_CUST_UN
RRN=2 RRN=4 RRN=5 RRN=6 RRN=7 RRN=8 RRN=9 RRN=10
>>Ret=8; Err=0; Msg=; Int=0
```

Le type d'opération de lecture est indiquée par « VERB=
Ou xxxx peut être :
FIRST, NEXT, PREV, LAST, KEY_EQ, KEY_GE, KET_GT, …

« **LOCK=ON/OFF** » indique si l'opération se fait avec ou sans verrou.
« **RECS=** » indique le nombre d'enregistrements maximum demandés en réponse.
Ce nombre est directement lié à l'information « Records= », section « Buffers » du fichier « Easycom.ini » sur le PC.
« **RRN=** » donne une information sur les numéros des enregistrements lus.

En réponse, « Ret=n » indique le nombre d'enregistrements effectivement retournés, suite à la demande de lecture.

Si la lecture échoue à cause d'une erreur d'Entrée/Sortie, le message est écrit dans la trace, et les enregistrements effectivement lus sont retournés.

```
<<EACread(2,p(3570),102,34144291,(null),0,p(140))
VERB=_EAC_NEXT LOCK=OFF RECS=35 FILE=EASYCOM/SP_CUST_UN
RRN=11 RRN=12 RRN=13 RRN=14 RRN=15 RRN=16 RRN=17 RRN=18 RRN=19
RRN=20 RRN=54 +...
... RRN=55 RRN=56
**SIGIO** Msg:CPF5001

>>Ret=13; Err=5001; Msg=CPF5001; Int=0
```

Dans cet exemple, 35 enregistrements sont demandés, 13 seulement sont disponibles avant la fin de fichier.

Pour obtenir le message détaillé de l'erreur, utilisez la commande DSPMSGD sur l'AS/400.

Trace d'une ouverture d'une requête SQL

```
 <<EACopen(SELECT * from SP_CUST where LASTNAME>'M',4194309,-1)
Statement : SELECT * from SP_CUST where LASTNAME>'M'
```

```
Cursor 0
>>Ret=2; Err=0; Msg=; Int=0
<<EACgetdesc(2,p(65000),65000,939786240,(null))
>>Ret=9; Err=0; Msg=; Int=0
```

Erreurs courantes

Erreur de connexion 10060 (274C Hexa) : Connection Time out

L'adresse TCP/IP contactée n'existe pas sur le réseau.

Vérifier l'adresse TCP/IP de l'AS/400 ou son nom.

Si un nom de machine AS/400 est utilisé, vérifier qu'il est bien référencé sur les serveurs DNS.

Si vous êtes sûr de l'adresse IP, c'est sûrement le fait d'un firewall.

Si vous avez un firewall qui protège votre AS/400, le port 6077 doit être ouvert.

Erreur de connexion 10061 (274D Hexa) : Connection Refused

Vérifier l'adresse IP, ou le nom de l'AS/400.

Vérifier que le sous-système EASYCOM a bien été démarré sur l'AS/400.

Lancer le sous-système par la commande :

```
STRSBS EASYCOM/EASYCOM
```

Si le sous-système est bien démarré, vérifier que le travail EASYCOMD tourne.

Sinon, démarrer le travail EASYCOMD par la commande :

```
STREACD EASYCOM
```

Ou, arrêter le sous-système, et le relancez.

Tester la connexion par l'outil de configuration ou d'administration de EASYCOM.

Si le travail EASYCOMD ne démarre jamais, consulter les éventuels messages générés par EASYCOM par les commandes suivantes :

```
DSPMSG EASYCOM/EACMSGQ
```

Ou

```
DSPPFM EASYCOM/LOGFILE
```

EASYCOM utilise par défaut le port numéro 6077.

Si ce numéro est déjà utilisé, configurer EASYCOM pour en utiliser un autre.

Easycom écrit des informations dans le fichier **LOGFILE** à chaque problème de connectivité TCP/IP, notamment.

- Coupure non programmé. Par exemple si l'utilisateur coupe son processus windows par le gestionnaire de tâche, ou coupure réseau franche.
- Partie PC plantée pendant plus de 2 minutes (cas comprenant également câble débranché par exemple). C'est le 'heartbeat timeout'.

Le temps de 120 secondes est réglable par CFGEAC.

Erreur de connexion 11001 (2AF9 Hexa) : Host not found

L'AS/400 peut être désigné par un nom sur le réseau TCP/IP, au niveau du DNS ou du fichier host. Cette erreur survient lorsque ce nom est utilisé à la place de l'adresse IP dans les paramètres de la connexion et qu'il n'est pas trouvé et associé à la bonne adresse IP.

Vérifier le nom de l'AS/400, le fichier host, le(s) serveur(s) DNS, ou utiliser une adresse IP de type xxx.xxx.xxx.xxx

Où se trouve le fichier Hosts ?

En général dans le répertoire C:\WINDOWS\system32\drivers\etc.

Ce fichier contient les correspondances des adresses IP aux noms d'hôtes. # Chaque entrée doit être sur une ligne propre. L'adresse IP doit être placée dans la première colonne, suivie par le nom d'hôte correspondant. L'adresse IP et le nom d'hôte doivent être séparés par au moins un espace.

De plus, des commentaires peuvent être insérés sur des lignes propres ou après le nom d'ordinateur. Ils sont indiqué par le

symbole '#'.

Par exemple :

194.206.10.1 main.as # serveur AS400 principal

194.206.10.2 test.as # serveur AS400 de test

194.206.10.100 serveur.info1

194.206.10.101 poste_x

....

Serveur DNS

Vérifier l'adresse du serveur DNS dans les propriétés de Protocole Internet (TCP/IP) de votre connexion réseau.

Erreur "dll Easycom non trouvée"

Assurez-vous que la dll Easycom (**eac3100as.dll)** se trouve dans un répertoire du "PATH" ou dans le répertoire de l'application.

En développement la dll devrait se trouver dans le répertoire Programmes de WinDev (par défaut C:\WinDev 10\Programmes) et en déploiement dans le répertoire de l'application mais elle peut également se trouver dans n'importe quel dossier du path.

Si vous avez plusieurs dlls, attention en cas de mise à jour notamment.

La meilleure solution est de copier la dll dans le "PATH". Pour cela, sélectionnez les propriétés du poste de travail, onglet Avancé et bouton "Variables d'environnement".

You have no free connexion on EASYCOM Serveur

Ce message apparaît lorsque le nombre de connexions fixé par la licence a été dépassé.

Pour vérifier les connexions (jobs) actives :

Sur l'AS/400, saisissez la commande 'WRKACTJOB' et vérifier le nombre de travaux en cours.

EASYCOM et EASYCOMD ne comptent pas comme travaux en cours.

Attention, si vous avez plusieurs connexions ou si vous utilisez les fonctions `HChangeConnexion` ou `HDécritConnexion`, elles vont créer des jobs en plus.

Catalogue des erreurs

Catalogue des erreurs Easycom For WinDev

SQL	1	Erreur SQL native, renvoyée par le System I
SIGNAL	2	Erreur native de type signal. Par exemple CPF9810 : bibliothèque non trouvée
MULTIREC	3	Erreur d'écriture différée
APPC	4	Erreur APPC (n'est plus utilisé)
INTERNAL	5	Erreur interne à Easycom : erreur de clé, ou opération non conforme
TCPIP	6	Erreur TCP/IP (connexion, …)
UNKNOWN	7	Erreur inconnue (impossible de déterminer la catégorie)
EACWD	8	Erreur spécifique à Easycom For WinDev.

Catégorie 1 - erreurs SQL

Consulter la documentation IBM pour le détail des messages SQL.

Catégorie 2 - erreurs signal (CPF...)

Le texte abrégé est renvoyé en plus de la valeur du CPF.

Les fonctions `ASErreurAide` et `ASErreurDonnees` proposent le texte complet et les variables de l'erreur.

Catégorie 3 - Erreur d'écriture différée

Cette erreur ne peut survenir que si la fonction ASPropriete est utilisée avec l'option CACHEDINSERT.

Catégorie 5 - Internal

Erreurs de Protection

0x0101	257	Vous ne pouvez pas ouvrir un autre fichier spécifié pour la démonstration
0x0102	258	La protection de la clé n'est pas valide
0x0103	259	Licence EASYCOM/DOS non compatible avec EASYCOM/OS400
0x0104	260	Clé non valide
0x0105	261	Plus de connexions disponibles
0x0106	262	EASYCOM/DOS non compatible avec EASYCOM/OS400
0x0107	263	La protection de la clé n'a pas été activée
0x0108	264	Plus de stations disponibles
0x0109	265	Pas de licence pour cette option
0x0401	1025	Entrez à nouveau le mot de passe

Erreurs Générales

0x0001	1	Erreur de paramètre
0x0002	2	Erreur d'allocation mémoire
0x0003	3	Le fichier n'a pu être ouvert

Erreurs Particulières

0x0201	513	Pas d'erreur détectée sur le fichier
0x0202	514	Le fichier existe déjà
0x0203	515	La définition du fichier ne peut être trouvé
0x0208	520	Write operation on virtual format
0x0207	519	Sélectionnez le bon format
0x0209	521	Format non trouvé
0x020A	522	Le paramètre NULL ne peut pas être converti
0x020C	524	In writelock mode, cannot lock the wrote record
0x020D	525	In writelock mode, the record was modified between the wrote and the lock operations
0x020E	526	

0x020F	527	ALCOBJ non réussi
0x0210	528	
0x0211	529	Timeout on pgm call

Catégorie 6 - TCP/IP

Vous trouverez ci-dessous une liste des codes erreurs TCP/IP.

Les codes positifs sont directement des codes utilisés par le système, tandis que les codes négatifs correspondent à des échecs de protocole Easycom.

Attention, les codes positifs changent en fonction de la plateforme utilisée (Windows, Linux, AIX, iSeries, …)

Un texte d'erreur doit accompagner le code, en français ou anglais pour les messages Easycom et dans la langue locale pour les autres. Dans certains cas le message contient du texte venant de l'AS/400, et donc dans la langue de l'AS/400.

La liste ci-dessous comprend la liste exhaustive pour les codes négatifs, ainsi que la plupart des codes positifs pour Windows.

-1	Erreur de soumission du travail Easycom. Pour résoudre ce problème vérifier la file de messages EACMSGQ, ainsi que l'historique de EASYCOMD.
-2	Echec de validation utilisateur/mot de passe. Soit l'utilisateur, soit le mot de passe sont incorrects ou expirés. Un texte spécifique à l'erreur (renvoyé par l'AS/400) doit indiquer la raison.
-4	Le travail soumis n'a pas répondu. Le travail soumis s'est probablement arrêté avant de pouvoir communiquer. Faire WRKOUTQ OUTQ(QEZJOBLOG) pour consulter les messages des travaux s'étant arrêtés en erreur.
-5	Le mot de passe est expiré. HOuvreConnexion permet de faire la resaisie du mot de passe ou alors activer les messages Easycom pour que le login Easycom prenne en charge le changement de mot de passe.
-6	Rejet interne. Ne devrait pas se produire : il doit y avoir soit un bug, soit une incompatibilité de versions non gérée.
-7	La liste de bibliothèques associée à l'utilisateur n'a pas pu être mise en place. La JOBD de l'utilisateur peut être erronée.
-8	Le profil SSO est expiré ou n'est pas supporté par EasycomD. Un login non SSO est requis pour pouvoir se connecter.
-9	Le serveur n'accepte pas les tickets Kerberos.
-10	Timeout à la lecture. Un timeout de lecture a été défini et atteint. Il est vivement recommandé de déconnecter puis reconnecter.
-11	L'utilisateur a annulé la boîte de dialogue de login Easycom. (ou bien la boîte de dialogue de changement de mot de passe).
-12	Connexion 'cassée'. La connexion a été coupée brutalement.
-13	Erreur lors de la négociation Kerberos
-14	Erreur Kerberos côté client
-15	Erreur Kerberos côté serveur
-16	Version OS/400 incompatible (généralement suite à tentative d'utilisation de Kerberos)
-17	Erreur inattendue pendant la soumission du job (état inconnu). Un évènement imprévu inconnu s'est produit pendant la soumission du travail Easycom.
-18	Authentification Kerberos hors plage horaire.
-19	Connexion hors plage horaire, décidée par exit program
-20	Connexion interdite, décidée par exit program
-21	Identification non reconnue, décidée par exit program
-22	Kerberos n'est pas supporté par le serveur (non installé ?)
-23	Authentification Kerberos obligatoire. La connexion Easycom par utilisateur/mot de passe est interdite sur ce système.
-24	Echec d'utilisation de la bibliothèque Easycom demandée.

-25	Echec de réveil d'un travail. L'application a tenté de réveiller un travail en attente, mais l'opération a échoué. Une nouvelle connexion est requise. Remarque : cette fonction n'est utilisée pour le moment que par Easycom For PHP
-26	SSL est requis sur le serveur. La négociation SSL n'a pas été configurée ou a échoué, mais est requise par le serveur. Elle a pu être configurée comme obligatoire au niveau de EACTCPP01 ou bien via la commande CFGEAC.
-27	Erreur SSL sur le serveur. La négociation SSL a échoué du fait d'une erreur détectée sur le serveur. Il doit y avoir des informations complémentaires dans la fille d'attente EACMSGQ (faire DSPMSG EASYCOM/EACMSGQ)
-28	La négociation SSL a fonctionné, mais un problème s'est produit lors du passage de la connexion vers le travail.
-29	Erreur SSL sur le client. La négociation SSL a échoué du fait d'une erreur côté client. Des informations supplémentaires sont disponibles dans le texte d'erreur associé.
-30	Erreur de séquence SSL. La négociation SSL a détecté une séquence invalide !
-31	Erreur de protocole SSL. Une erreur s'est produite dans le protocole SSL (poignée de main, …)
-32	Erreur SSL : type d'interface SSL non supportée sur la plateforme
-33	Mappage EIM obligatoire (connexion par utilisateur non acceptée)
-34	Authentification SSL (par certificat client) obligatoire
-35	Erreur d'authentification par certificat. Le certificat peut être invalide, expiré, inexistant, clé privée non disponible, …
-36	Erreur de mappage EIM.
-37	Aucune méthode d'authentification valide n'a été reçue.

Voici les erreurs TCP/IP les plus courantes:

10061	Connexion refusée. Aucune connexion n'a pu être établie car la machine cible les refuse. Cela signifie généralement que le service sur la machine cible n'est pas actif. Un firewall peut également provoquer ce type d'erreur.
10060	Timout à la connexion. La tentative de connexion a échoué car la machine n'a pas répondu dans le temps imparti. Le service cible peut être indisponible, ou un firewall peut avoir bloqué la communication.
11001	Hôte non trouvé. L'hôte n'a pas été trouvé. Le nom d'hôte n'est pas un nom ou un alias reconnu, ou n'a pas pu être trouvé dans les bases de données (DNS) contactées.
10053	Connexion abandonnée. La connexion a été abandonnée par le serveur, probablement à la suite de timeout atteints. Cela peut être aussi suite à un passage hors portée lors de l'utilisation de réseaux sans fil. Un firewall peut également déclencher cette erreur (il autorise la connexion, mais la coupe juste après).
10064	L'hôte est arrêté. L'hôte de destination n'est pas joignable car la machine a été arrêtée, ou n'est plus disponible
10050	Le réseau est arrêté. Le réseau local ne répond plus, par exemple parce qu'aucune carte réseau n'est active.

Remarque : la plupart des problèmes de connexion sont dues à des firewall présents sur les postes utilisateur ou au niveau des routeurs.

Easycom utilise le port tcp 6077 part défaut.

Catégorie 8 - erreurs Easycom WinDev

1	**EAC_ERR_ERROR**	**Une erreur est survenue**
2	**EAC_ERR_TOOMUCHOPENFILE**	**Trop de fichiers ouverts**

3	EAC_ERR_MEMALLOC	Mémoire disponible insuffisante
4	EAC_ERR_INVALIDPTR	Pointeur invalide (handle)
5	EAC_ERR_FILENOTFOUND	Fichier non trouvé
6	EAC_ERR_FIELDNOTFOUND	Champ non trouvé
7	EAC_ERR_INVALIDFIELDNBR	Le numéro du champ n'existe pas
8	EAC_ERR_INVALIDKEYLEN	La taille de la clé est invalide
9	EAC_ERR_INVALIDKEYNBR	Le numéro de la clé est invalide
10	EAC_ERR_NOTENABLETOUPDATE	Le fichier ne peut être modifié
11	EAC_ERR_INVALIDOPENMODE	Le type d'ouverture du fichier ne permet pas ce type d'opération
12	EAC_ERR_RECORDNOTFOUND	Enregistrement non trouvé
13	EAC_ERR_RECORDLOCKED	L'enregistrement est bloqué
14	EAC_ERR_BEOF	Fin ou début de fichier
15	EAC_ERR_FILELIMITS	Tentative de lecture en dehors du fichier (avant ou après EOF)
16	EAC_ERR_NOTCONNECTED	Pas connecté
17	EAC_ERR_INVALIDSEQ	Séquence d'opération non valide.
18	EAC_ERR_NORANGESET	Intervalle non définie.
19	EAC_ERR_NOLINKDEFINED	Lien non défini
20	EAC_ERR_NOCURRENTRECORD	Aucun enregistrement courant
21	EAC_ERR_NULLNOTALLOWED	Les opérations sur NULL ne sont pas applicables (le champ ne supporte pas la valeur NULL)
22	EAC_ERR_BADSESSION	La session avec l'AS/400 n'existe pas.
23	EAC_ERR_WRONGLOGIN	Erreur dans le nom d'utilisateur ou le mot de passe.
24	EAC_ERR_NOTENOUGHRIGHTS	Problèmes avec les droits de l'utilisateur.
25	EAC_ERR_INVALIDTYPE	Type non valide.
26	EAC_ERR_INVALIDINFO	La taille de la variable de récupération des informations n'est pas assez grande.
27	EAC_ERR_NOTTYPEPROPERTY	La propriété demandée n'existe pas.
28	EAC_ERR_RECORDCHANGED	L'enregistrement a changé avant son actualisation.

29	**EAC_ERR_ALLREADYINTRAN**	**Déjà dans le mode transactionnel.**
30	**EAC_ERR_NOTINTRAN**	**Aucune transaction commencée.**
4097	**EAC_ERR_FIELDNULL**	**La valeur du champ est NULL**

1000	Paramètres incorrects
1001	Handle de connexion retourné par l'AS/400 est négatif
1002	Tentative de lecture du fichier sans l'avoir ouvert
1003	Impossible d'allouer de la mémoire pour ouvrir un objet NAAccesTableAS400
1004	Absence de connexion
1005	Problème pour récupérer le nombre de champs
1006	Nombre de champs différents
1007	Nombre de clé différent
1008	Aucun nom de champs de la table AS/400 ne correspond a celui du fichier PC
1009	Impossible de créer le fichier de la trace SQL
1010	Différence entre l'indice du champ AS/400 et du champ WinDev
1011	Impossible de placer l'enregistrement en mode edit
1012	Impossible de récupérer la valeur du champ
1013	Le bookmark retourné est invalide
1014	Erreur de lecture d'une ligne
1015	Impossible d'ouvrir le fichier AS/400, erreur inconnue
1016	Impossible de fixer la borne de recherche
1017	La recherché est impossible
1018	La date n'a pas été formatée de manière correcte
1019	Problème dans la comparaison de la clé
1020	Problème d'ouverture du fichier AS/400
1021	Impossible de créer la table sur l'AS/400
1022	Impossible d'ouvrir le fichier
1023	Impossible de se positionner sur la première ligne de la table ouverte
1024	Impossible de récupérer la longueur AS/400 du champ
1025	Impossible de fermer le fichier
1026	Impossible de détruire la référence de la clé dans la table de Hash
1027	Erreur sur le déplacement vers un bookmark
1028	Erreur sur restaure position
1029	Impossible d'obtenir le nombre d'enregistrements
1030	Erreur sur l'ajout d'un enregistrement
1031	Erreur de filtre (syntaxe)
1032	Erreur de filtre (syntaxe)
1033	Erreur sur l'ouverture d'un filtre (syntaxe)
1034	Impossible d'ouvrir le fichier mémo
1035	Erreur d'intégrité du fichier mémo
1036	Erreur d'écriture de mémo
1037	Erreur de lecture sur le fichier mémo à attacher
1038	Erreur de lecture sur le fichier mémo à extraire
1039	System memory allocation failed

1040	Clé en double
1041	Erreur de modification
1042	Erreur intégrité type incompatible
1043	Ouverture : incompatibilité de structure (au sens large)
1044	Pas d'index
1045	Opérateur de filtre inconnu
1046	Valeur incorrecte dans le filtre
1047	Valeur inconnue dans le filtre
1048	Nom de champ inconnu dans le filtre
1049	Opération non supportée
1050	Informations étendues ou fichiers d'alias non renseignées pour une clé : corriger ou utiliser l'option IGNORE_EMPTY_EXTINFO
1051	Problème d'initialisation de l'Accès Natif. Vérifier la compatibilité du programme appelant l'Accès Natif AS/400
1052	La licence de déploiement n'est pas valide pour utiliser WinDev Pocket PC. Vous devez demander une extension de licence pour pouvoir déployer

Exporter une analyse (constructeur DDS)

Introduction

Le constructeur de DDS est un outil d'exportation de fichiers présents dans une analyse WinDev vers des fichiers AS/400.

Il s'agit d'un programme écrit en WinDev et qui utilise évidemment EASYCOM.

Par défaut il est installé dans le répertoire " C:\Program Files (x86)\Easycom\WinDev 2026\ExportAS400".

La fonctionnalité Génération de script permet de créer un répertoire avec tous les fichiers nécessaires pour générer les fichiers depuis un autre poste (chez le client final notamment) sur un autre AS/400.

Pour le lancer : Démarrer - Programmes - Easycom For WinDev 2026 - Exportation de fichiers vers l'AS/400.

Son utilisation se fait en deux temps :

- Ouverture d'une analyse et préparation des données (connexion facultative)
- Génération des fichiers (directe ou par script, connexion nécessaire).

Connexions - menu Serveur

La connexion n'est nécessaire que pour la génération des fichiers, vous pouvez utiliser le mode Hors Connexion (menu Serveur) pour préparer votre analyse, modifier les noms, associer les types, définir les bibliothèques et autres options.

Le profil utilisé pour accéder à l'AS/400 doit évidemment **avoir les droits pour écrire et compiler dans les bibliothèques spécifiées.**

Si vous utilisez plusieurs AS/400, le menu Serveur vous propose la liste des dernières connexions utilisées. Dans le menu Outils - Paramétrages - Environnement vous pouvez de plus choisir de sauvegarder les informations de connexion et activer une connexion automatique au démarrage.

Analyse

Le constructeur de DDS utilise nécessairement une analyse (fichier au format *.wdd) : il n'est pas possible de sélectionner et d'exporter un fichier Hyper File isolé.

Aucune modification n'est effectuée sur l'analyse elle-même, les fichiers exportés conservent leurs propriétés d'origine. Pour utiliser ensuite ces fichiers dans un projet WinDev ils doivent ensuite être importés ou synchronisés.

L'analyse peut contenir des fichiers qui sont de type AS/400 (avec des infos étendues EASYCOM qui peuvent être récupérées - option dans le menu 'Outils' 'Paramétrages' – Onglet 'Valeurs par défaut' – (Utilisation des 'infos étendues'), **toute modification doit être suivie par une synchronisation des fichiers.**

Le constructeur de DDS sauvegarde les modifications faites sur une analyse et conserve un historique pour y accéder directement.

Remarque : il est possible d'exporter tous les fichiers présents dans l'analyse, y compris des fichiers de bases externes (MySQL, Access...).

Fichier source

La structure d'un fichier de données AS/400 (fichier physique) et de ses index (fichiers logiques) est différente de celle d'un fichier WinDev. Les descriptions de fichiers sont stockées dans un fichier source (de type QDDSSRC), chaque membre de ce source contient la description d'un fichier et il doit être compilé pour créer les fichiers eux-mêmes.

Si le fichier source n'existe pas, le constructeur de DDS peut le créer si la directive "Créer la source DDS" est bien cochée.

Directives

A définir au niveau global (Paramétrages - Valeurs par défaut) ou pour chaque fichier, les directives précisent les opérations à effectuer : création et compilation du source, création des liaisons, transfert des données et journalisation..

Cet outil permet donc à partir d'une analyse WinDev de:

- générer une fichier source avec les structures de fichiers sur l'AS/400 (Fichiers physiques "PF" et logiques"LF")
- compiler le fichier source pour créer les fichiers,
- créer des contraintes référentielles (liaisons) simples,
- démarrer la journalisation du fichier,
- transférer les données.

Fichiers créés

Le constructeur de DDS crée de façon transparente pour chaque fichier Hyper File, un ou des fichiers sur l'AS/400 :

- (un fichier source de type QDDSSRC)
- un fichier physique qui contient les données (et éventuellement une clé),
- un fichier physique pour stocker les éventuels mémos (voir mémos),
- un fichier logique associé au physique pour chaque rubrique de type clé,

Les fichiers logiques peuvent être référencés en *LIBL (le nom de la bibliothèque du physique n'apparaît pas dans le source).

Noms

Dès l'ouverture un contrôle est fait sur les noms afin de garantir l'unicité de chaque objet et de limiter la taille des noms systèmes à 10 caractères. Des chaînes de transformation personnalisables permettent un contrôle précis des noms de tous les objets.

Type des données

Un des aspects essentiels du constructeur est de gérer les associations entre les types de données WinDev et AS/400 afin de garantir la cohérence des données, il est possible de choisir par défaut une association stricte, commune ou l'utilisation des masques de saisie (formats) de chaque rubrique de manière globale ou de personnaliser pour chaque rubrique une association particulière.

Paramétrages

Le constructeur de DDS propose un ensemble de paramètres communs à définir avant l'ouverture de l'analyse, la plupart sont les options et propriétés par défaut que l'on retrouve au niveau de chaque fichier ou clé (nom du source, bibliothèque, référencement, directives...)

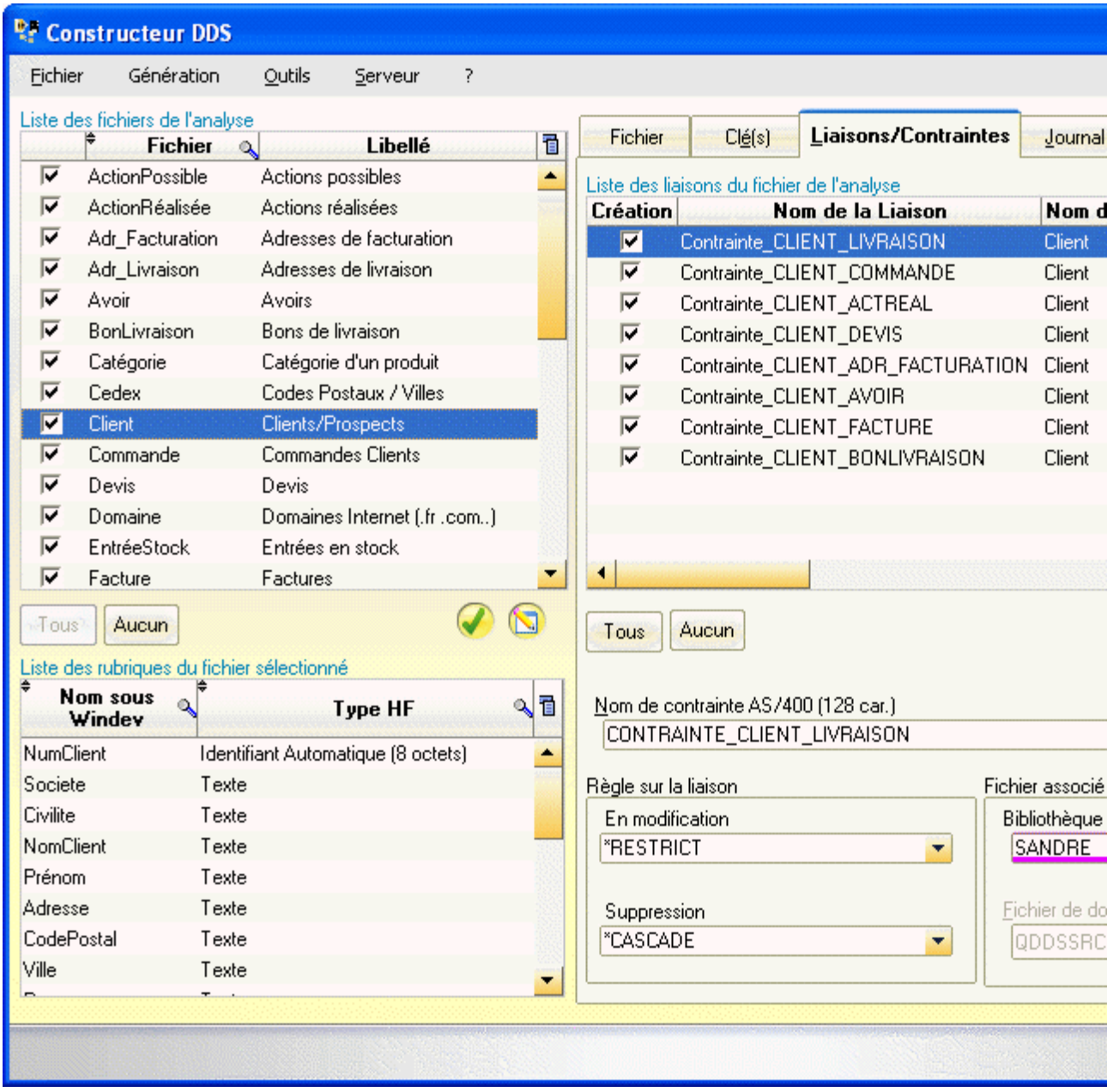
Il propose également quelques options plus particulières et spécifiques.

- la sélection d'une séquences de tri : par défaut National (*LANGIDUNQ),
- par défaut un WAITRCD(*IMMED) pour les verrouillages et lectures bloquantes,
- accepter la valeur nulle par défaut,
- utiliser le binaire 8,

Présentation générale

Commencer par ouvrir l'analyse où se trouvent les fichiers à exporter (menu Fichier - Ouvrir une analyse ou Ctrl - O). Il est possible de sauvegarder les analyses et de bénéficier ainsi d'un accès et d'une ouverture directe par le menu Fichier.

Il est vivement recommandé de **définir les valeurs par défaut (menu Paramétrage) avant d'ouvrir l'analyse** et d'éditer un fichier ou une rubrique.



La partie de droite se compose de 4 onglets, actifs ou pas en fonction des options et des propriétés du fichier sélectionné (en bas à droite de l'onglet Fichier).

- Fichier
- Clé(s)
- Liaisons/Contraintes
- Journal

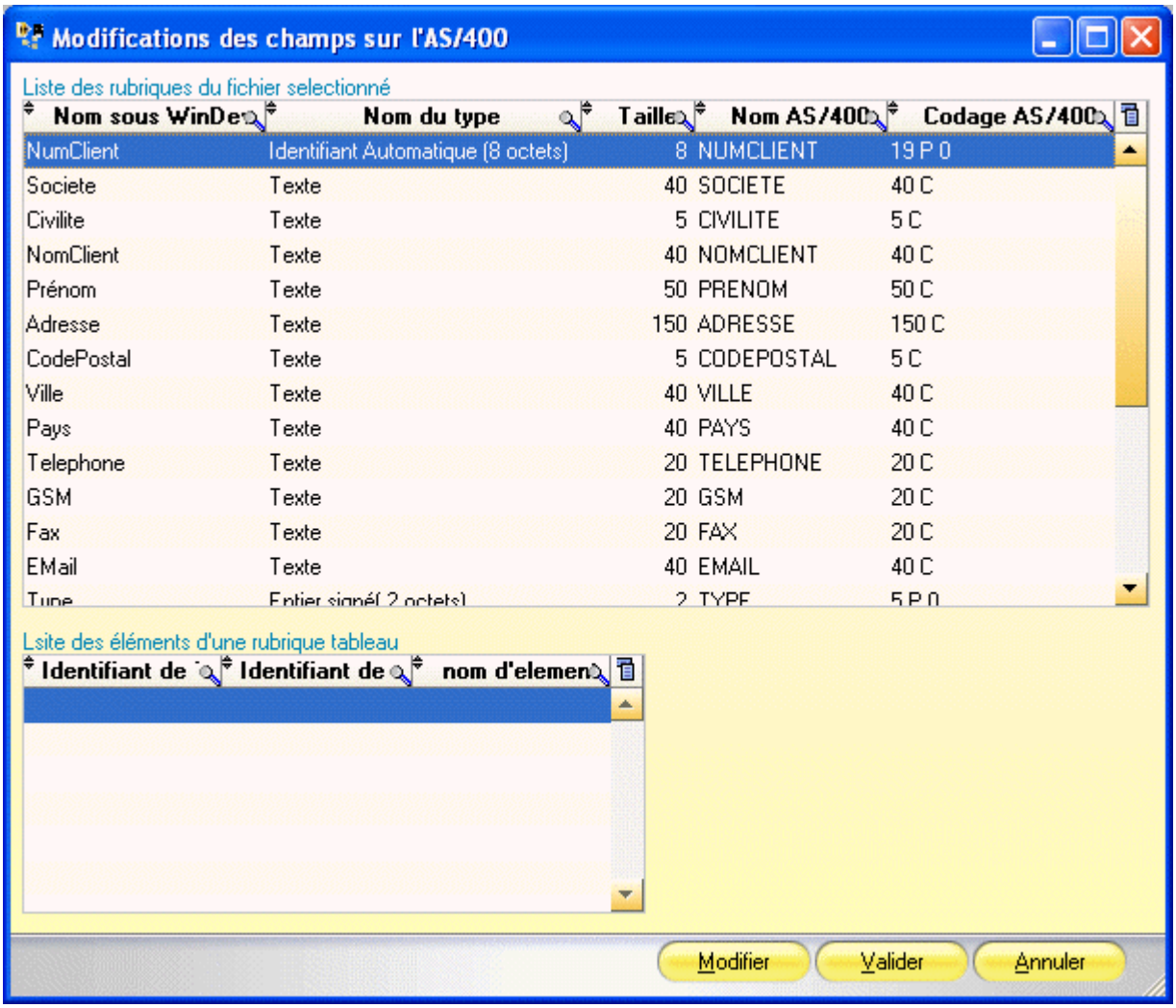
Le cadre du haut propose la liste des fichiers de l'analyse, pour chaque fichier sélectionné les rubriques qui le composent apparaissent dans le cadre en bas.

Il s'agit ici de sélectionner les fichiers de l'analyse que l'on souhaite exporter :

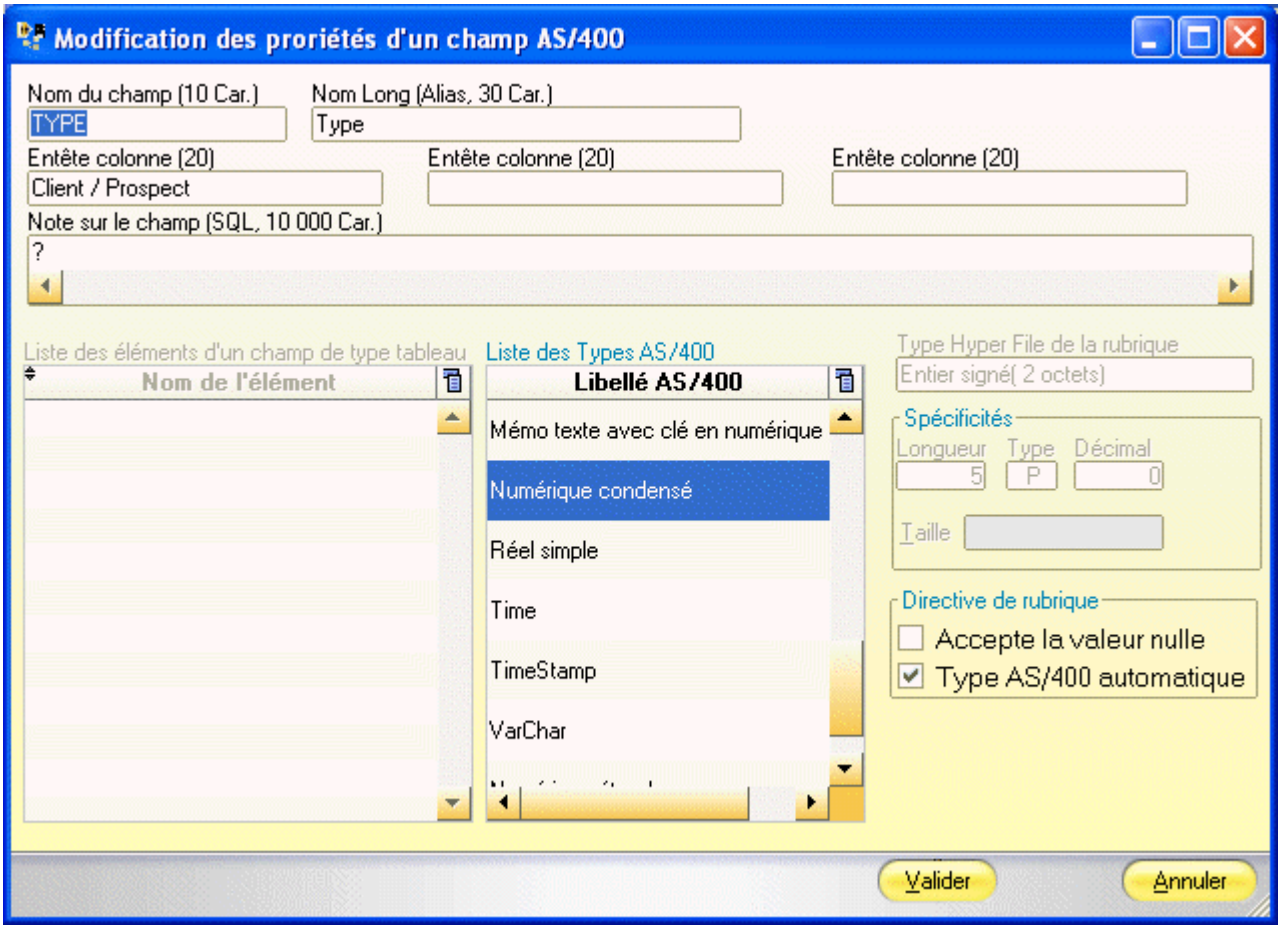
Tous — Sélectionner tous les fichiers de l'analyse

Aucun	Désélectionner tous les fichiers de l'analyse
	Consulter la liste des champs du fichier sélectionné
	Rafraîchir la liste avec uniquement les fichiers sélectionnés
	Afficher tous les fichiers

Double-cliquer sur un fichier pour consulter le détail des rubriques dans une fenêtres distincte.



Double-cliquer sur une rubrique pour éditer ses propriétés.



Il est possible d'autoriser par défaut pour toutes les rubriques la valeur Null depuis le paramétrage ou pour une rubrique particulière dans cet écran.

Si Type AS/400 automatique est coché, la liste des types AS/400 est désactivée (voir associations de types).

Changer les associations de type

Menu 'Outils', 'Changer les associations de type / Appliquer les associations'.

Un aspect essentiel du constructeur de DDS, et en général de l'accès natif AS/400, est bien évidemment de gérer des types de données différents entre un fichier Hyper File et AS/400. Si les types les plus courants ont leur équivalent (Chaîne en CHAR ou VARCHAR, entiers...) certains formats sont plus délicats (dates, heures, monétaires, mémos...).

Il y a également des considérations de taille à prendre en compte sur des utilisations "limite", ainsi WinDev propose des chaînes jusqu'à 64 ko mais le format CHAR de l'AS/400 est limité à 32Ko.

Attention, toutes les associations n'ont pas forcément un sens et peuvent générer des erreurs au moment de l'accès aux données.

Types AS/400

Les différents types sont résumés dans le tableau ci-dessous. En plus des types normaux, la gestion des mémos et le lien entre le fichier principal et le fichier des mémos peut avoir différentes formes.

Chaînes	Numérique	Mémos
C Char CH Char Hexa VC VarChar	P Numérique condensé Z Numérique étendu S Réel simple DB Réel double I1 Entier sur 1 octet I2 Entier sur 2 octets I4 Entier sur 4 octets I8 Entier sur 8 octets	MBB Mémo binaire avec clé binaire MBP Mémo binaire avec clé en numérique condensé MBZ Mémo binaire avec clé en numérique étendu MTB Mémo texte avec clé binaire MTP Mémo texte avec clé en numérique condensé MTZ Mémo texte avec clé en numérique étendu
Dates et heures D Date T Time TS TimeStamp		

Types WinDev

0	type inconnu
1	identifiant automatique sur 8 octets
2	rubrique texte
3	rubrique de type entier signé sur 2 octets
4	rubrique de type entier signé sur 1 octet
5	rubrique de type entier signé sur 4 octets
6	rubrique de type réel simple (sur 4 octets)
7	rubrique de type réel double (sur 8 octets)
8	rubrique de type numéro d'enregistrement (entier non signé)
9	rubrique de type entier non signé sur 2 octets
10	rubrique de type date sur 6
11	rubrique de type heure
12	rubrique de type entier non signé sur 1 octet
13	rubrique de type réel turbo
14	rubrique de type date sur 8
15	rubrique de type mémo texte
16	rubrique de type mémo binaire 4.0
17	rubrique de type monétaire
18	rubrique de type mémo binaire, mémo Image, mémo Son, mémo OLE ou mémo binaire autre
19	rubrique de type entier signé sur 8 octets
20	rubrique de type entier non signé sur 8 octets
21	rubrique de type image
22	rubrique de type entier non signé sur 4 octets
23	rubrique de type chaîne binaire
24	rubrique de type date+heure
25	rubrique de type durée
26	rubrique de type caractère
27	rubrique de type booléen
28	rubrique de type identifiant automatique (sur 4 octets)

Afin d'associer la rubrique à son type d'origine au moment de l'importation, le type Hyperfile d'origine est stocké dans la DDS pour chaque rubrique :

```
A IDCLIENT 9P 0B COLHDG('Identifiant de' 'Client')
TEXT('HFTYPE=28')
A NOM 64A B COLHDG('Nom')
TEXT('HFTYPE=2')
```

Au moment de l'importation ce sont donc toujours les types Hyper File de l'analyse d'origine qui sont attribués, même si des associations particulières sont utilisées.

Mémos

Le constructeur de DDS ne crée pas de champs BLOB pour les mémos binaires mais un fichier qui reprend les 8 premiers caractères du nom du fichier d'origine suivi de deux underscores (__).

Ce fichier contient des blocs de données binaires ou hexadécimales et sa description de ce fichier ressemble à :

```
A* DDSBLD : MEMO DU FICHIER HYPERFILE 'CLIENT '
A UNIQUE
A R RCLIENT
A IDMEMO 9B 0B COLHDG('NUMERO DU MEMO')
A IDBLOC 4B 0B COLHDG('NUMERO DE BLOC')
A TAILLE 9B 0B COLHDG('Taille totale')
A DONNEE 512H B VARLEN COLHDG('Données')
A K IDMEMO
A K IDBLOC
```

Chaque enregistrement du fichier parent contient un champ mémo avec un identifiant, dans le fichier mémo cet identifiant est associé à des blocs de données (512 octets par défaut)

Le constructeur de DDS propose différent types pour les identifiants de mémo dans le fichier principal. Il n'est en général pas nécessaire de modifier ces types à moins que d'autres programmes de l'AS/400 (RPG) soient susceptibles d'utiliser ces données.

En revanche l'importation d'un fichier AS/400 (voir import DDS) qui contient des blobs va bien lui associer des mémos et gérer une correspondance directe (la journalisation est nécessaire dans ce cas).

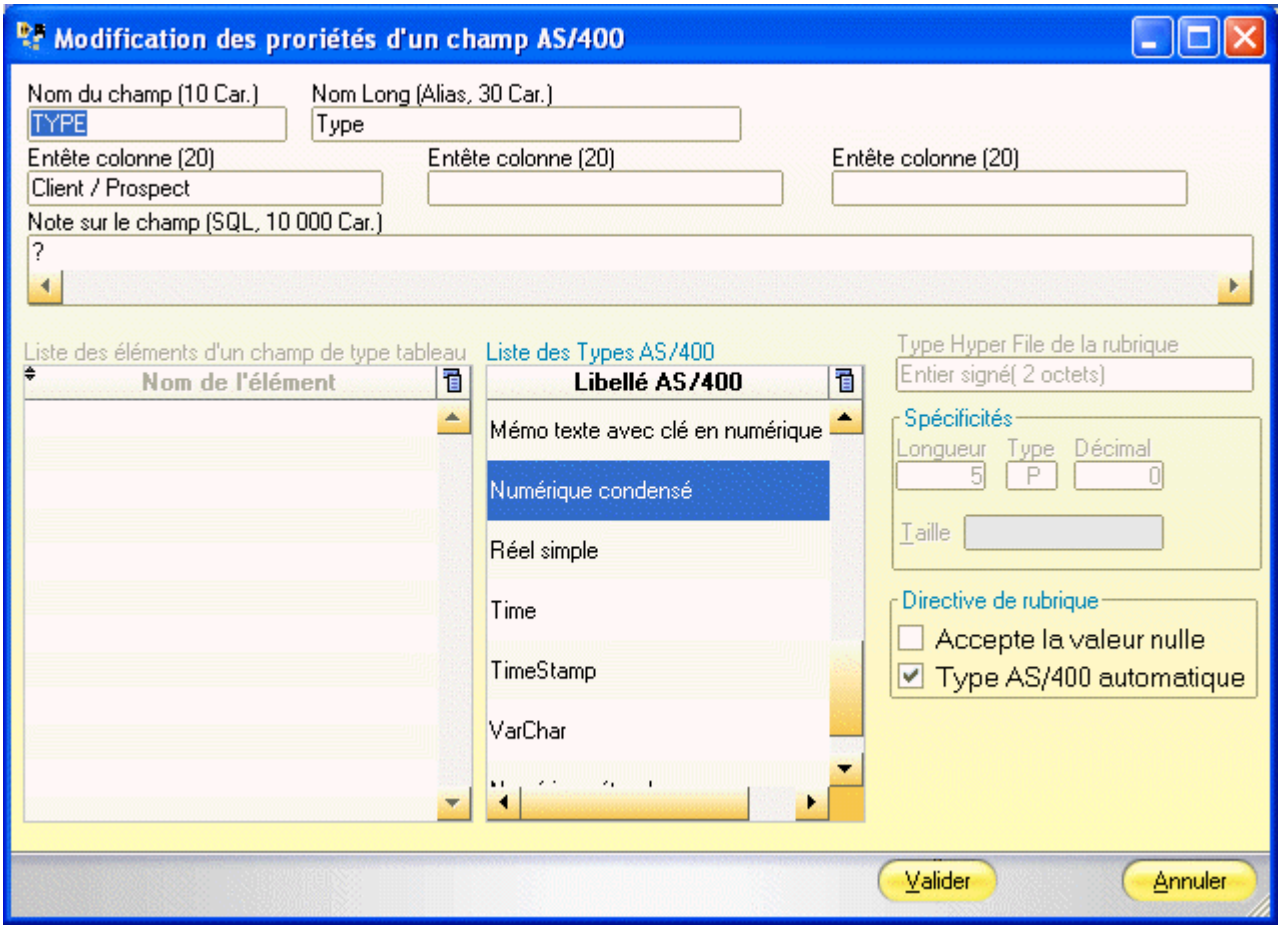
Initialisation stricte ou commune

Les associations de types font le lien entre un format WinDev et AS/400. Par défaut le constructeur vous propose un initialisation commune (les types les plus courants) ou stricte (types identiques autant que possible).

Type WinDev	COMMUNE	STRICTE
Booléen	Numérique condensé	Entier sur 1 octets
Caractère	Char	Char
Chaîne binaire	Char Hexa	Char Hexa
Date (6 digit)	Date	Date
Date (8 digit)	Date	Date
Date + Heure	TimeStamp	TimeStamp

Durée	TimeStamp	TimeStamp
Entier non signé (8 octets)	Numérique condensé	Entier sur 8 octets
Entier non signé (1 octet)	Numérique condensé	Entier sur 1 octet
Entier non signé (2 octets)	Numérique condensé	Entier sur 2 octets
Entier non signé (4 octets)	Numérique condensé	Entier sur 4 octets
Entier signé (8 octets)	Numérique condensé	Entier sur 8 octets
Entier signé (1 octet)	Numérique condensé	Entier sur 1 octet
Entier signé (4 octets)	Numérique condensé	Entier sur 4 octets
Entier signé (2 octets)	Numérique condensé	Entier sur 2 octets
Heure	Time	Time
Identifiant Automatique (8 octets)	Numérique condensé	Entier sur 8 octets
Identifiant automatique (4 octets)	Numérique condensé	Entier sur 4 octets
Image	Mémo binaire avec clé en numérique condensé	Mémo binaire avec clé en numérique condensé
Monétaire	Numérique condensé	Numérique condensé
Mémo binaire	Mémo binaire avec clé en numérique condensé	Mémo binaire avec clé en numérique condensé
Mémo texte	Mémo Texte avec clé en numérique condensé	Mémo Texte avec clé en numérique condensé
Numéro d'enregistrement	Numérique condensé	Numérique condensé
Réel double	Numérique condensé	Réel double
Réel simple	Numérique condensé	Réel simple
Texte	Char	Char

Vous pouvez personnaliser une association pour un type en particulier, de manière globale depuis le menu Outils - Changer les associations de type puis l'appliquer par le menu Outils - Appliquer les associations de types à l'analyse ouverte mais il est également possible de préciser le type d'une rubrique en double-cliquant dessus et en sélectionnant un autre type dans la liste (décocher l'option Type AS/400 automatique si la liste est désactivée).



Utilisation du masque de saisie

Enfin il est possible de créer les types en reprenant le format de la rubrique lorsque l'option est coché dans les associations de types (Outils - Changer les associations de types).

Par exemple, une rubrique monétaire avec un format 999.99 de type Réel sur 4 octets sera exportée avec un format de 3P2 au lieu de 5P2.

Noms des fichiers et des clés

La question des noms est centrale, un fichier Hyper File va correspondre à plusieurs fichiers sur l'AS/400, chaque clé du fichier étant associée à un fichier logique. Lorsque plusieurs fichiers Hyper File comportent des clés avec des noms identiques (ce qui est courant) il est donc nécessaire de renommer les fichiers logiques.

Par ailleurs les noms de plus de 10 caractères doivent correspondre à un nom d'objet court, sur 10 caractères maximum.

Le constructeur propose une transformation automatique des noms, selon les règles ci-dessous, et la possibilité de personnaliser ces conversions par des chaînes spéciales.

Unicité des noms

Le constructeur de DDS effectue ce contrôle dès l'ouverture de l'analyse et propose des nouveaux noms construits sur le nom du fichier, de la rubrique et une numérotation. Ces noms sont paramétrables par les chaînes de transformation.

Les noms longs (jusqu'à 30 caractères) sont cependant conservés et stockés dans les alias SQL, ils seront ainsi récupérés lors de l'importation des DDS.

Par exemple une même clé "IDClient" dans un fichier "Client" et un fichier "Facture" donnera deux logiques distincts :

CLIIDCLI01 pour le logique du fichier Client.

FACIDCLI01 pour le logique du fichier Facture.

Dans l'analyse, après importation ou synchronisation des fichiers, cette information se retrouve dans les infos étendues de la rubrique (clé) sous la forme suivante :

Infos étendues de la clé IDClient du fichier Client :

```
<EASYCOM>
LF=
</EASYCOM>
```

Infos étendues de la clé IDClient du fichier Facture :

```
<EASYCOM>
LF=
</EASYCOM>
```

Longueur des noms système

Les noms de plus de 10 caractères sont exportés sous la forme d'alias SQL mais le nom système doit rester sur 10 caractères au plus (on peut limiter encore la taille des fichiers et des champs depuis le menu Outils - Paramétrages - Valeurs globales).

Par défaut, le nom de l'objet est créé selon les règles ci-dessous :

Fichiers physiques

- 8 premiers caractères du nom du fichier Hyper File (XXXXXXXX)
- 2 chiffres correspondant au numéro du fichier (YY)

La chaîne "XXXXXXXXYY" va transformer "FOURNISSEUR" en "FOURNISS01".

Fichiers logiques

Le logique reprend le nom de la rubrique, si ce nom fait plus de 10 caractères il sera composé des :

- 3 premiers caractères du nom du fichier physique : WWW
- 5 premiers caractères du nom de la clé XXXXX
- 2 chiffres correspondant au numéro de la clé YY

La chaîne "WWWXXXXXYY" va transformer la rubrique clé Numéro du fichier Fournisseur en FOUNUMER01.

Rubriques

Un nom de rubrique qui dépasse 10 caractères

- 8 premiers caractères du nom du champ (XXXXXXXX)
- 2 chiffres correspondant au numéro du fichier (YY)

La chaîne "XXXXXXXXYY" va transformer "IdentifiantClient" en "IDENTIFI01".

On peut lui préférer une chaîne de transformation telle que "XXZZZZZZZZZZXXXXXXYY" qui va transformer "IdentifiantClient" en "IDCLIENT01".

Chaîne de transformation

Les chaînes de transformation permettent de personnaliser les conversions de noms pour les fichiers et rubriques de plus de 10 caractères. On peut les éditer depuis le menu Outils - Paramétrages - Valeurs par défaut.

Z : caractère non retenu (fichier ou rubrique)

X : caractère retenu (fichier ou rubrique)

W : reprend le caractère du nom de fichier dans le nom de la rubrique

U : caractère du nom de fichier non retenu

Y : caractère remplacé par un chiffre (automatique)

[...] : les crochets permettent d'insérer un caractère

Exemples :

Nom du fichier	Nom de la rubrique	Chaîne	Résultat
Factures_clients		XXXZZZZZZXXXYY	FACCLI01
Factures_fournisseurs		XXXZZZZZZXXXYY	FACFOU01
Factures_clients	Adresse	WWWXXXXXYY	FACADRES01
Factures_clients	Adresse	WWW[_]XXXXYY	FAC_ADRE01
Factures_clients	Code postal	WWWUUUUUWWXXXYY	FACCLCOD01

Trois chaînes sont proposées :

- pour le nom du fichier (codes possibles : Z, X, U et Y)
- pour le nom du logique (clé) avec (Z, X, W, U et Y)
- pour les autres rubriques (Z, X et Y)

Au moment de l'importation

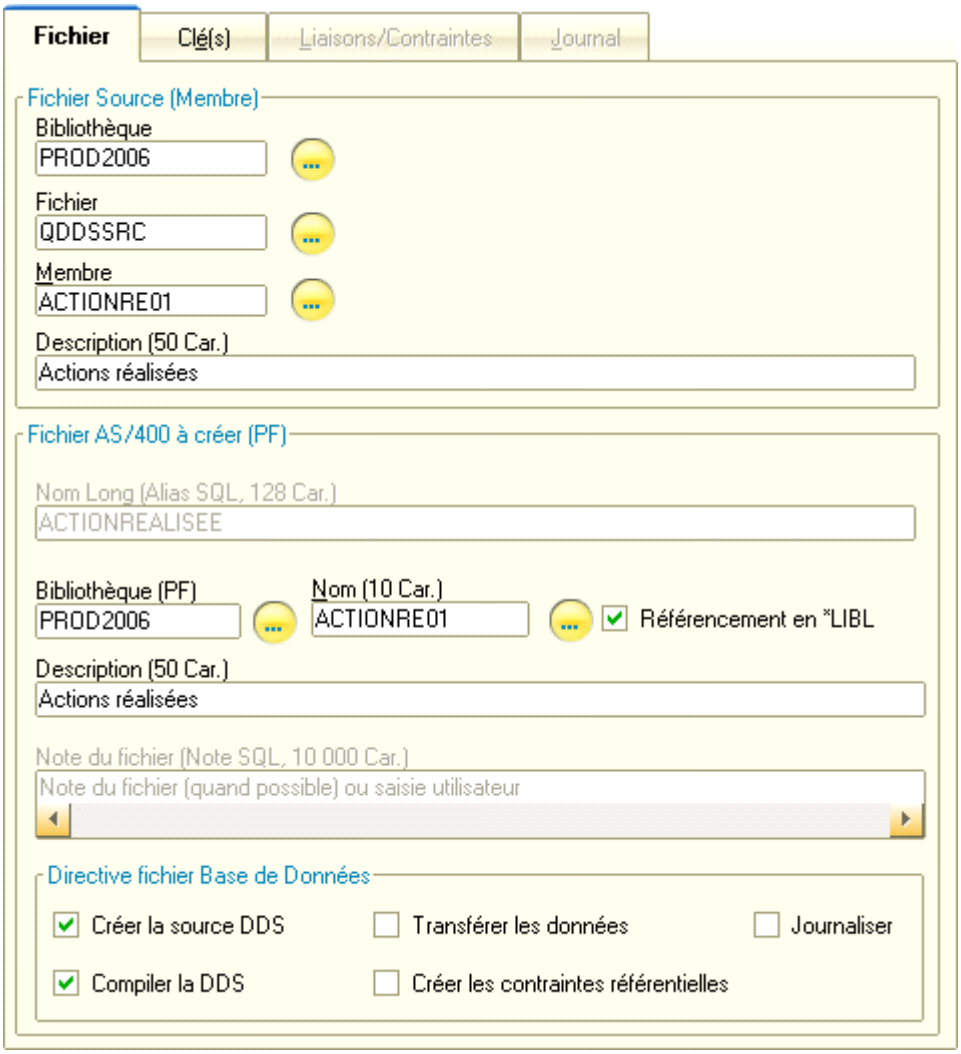
Lorsque les fichiers sont importés dans l'analyse, les noms longs (stockés dans les alias) sont restitués.

Depuis la version 9, les noms de fichiers sont importés par défaut en minuscules, si vous souhaitez contrôler le format du nom des fichiers et des rubriques, utiliser l'option LITERALCASE des infos étendues de la connexion (à définir avant l'importation).

Onglet 'Fichier'

Pour avoir un fichier source et une bibliothèque communs à tous les fichiers, les définir dans le menu Paramétrages - Valeurs par défaut. Si l'option "Auto" de ce même menu n'est pas cochée, toute modification de la bibliothèque ou du source pour un fichier en particulier sera proposée pour l'ensemble des fichiers.

Les différentes propriétés et directives au niveau de chaque fichier (à l'exception du nom du membre, du nom du fichier et de sa description) peuvent être définies globalement et sont explicitées dans la rubrique Paramétrage .

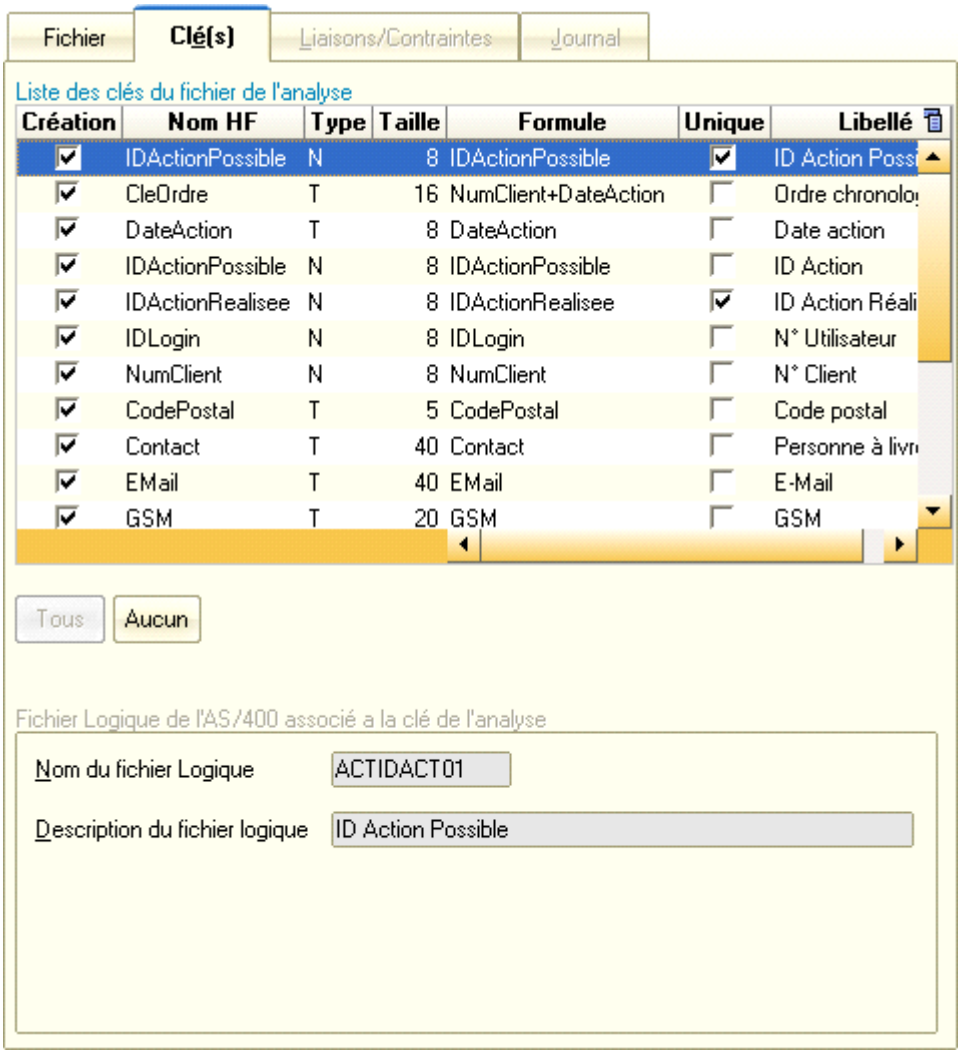


Les boutons proposent les listes des objets (bibliothèques, fichiers, membres) récupérées de l'AS/400, ces listes dépendent du profil utilisé et de sa LIBL.

Onglet 'Clé(s)'

Le principe essentiel est que chaque clé d'un fichier Hyperfile correspond à un fichier logique sur l'AS/400. Ce fichier logique doit être unique et avec un nom de 10 caractères au plus.

Un contrôle est fait dès l'ouverture de l'analyse.



Il est possible de modifier le nom et la description du fichier.

Il ne faut pas retirer le paramètre *HFQ pour une clé composée.

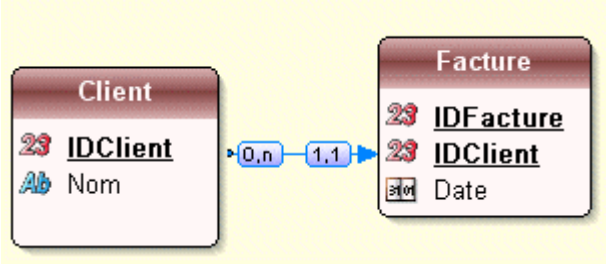
Onglet 'Liaisons/Contraintes'

Toutes les liaisons ne sont pas des contraintes.

Le constructeur de DDS ne permet d'exporter que les liaisons de type 0,n à 0,1 ou 1,1. Il proposera de convertir les liaisons 0,n à 0,1 ou 1,1 en liaisons 1,n à 0,1 ou 1,1.

Les autres liaisons sont des relations entre fichiers mais sans avoir le caractère de contrainte.

C'est le type de liaison le plus courant, par exemple une liaison sur la clé IDClient d'un fichier Client et d'un fichier Facture. Chaque client peut avoir 0, une ou plusieurs factures et chaque facture a un unique client.



Les attributs des clés doivent bien sûr être identiques.

Cette liaison se retrouve dans le constructeur avec un nom par défaut composé du nom du fichier parent suivi du fichier dépendant : CONTRAINTE_CLIENT_FACTURE.

Sur l'AS/400 cette contrainte sera associé au fichier dépendant (Facture) bien qu'elle apparaisse dans l'onglet du fichier parent dans le constructeur.

Il s'agit d'une contrainte de type *REFCST, c'est à dire une contrainte référentielle.

Certaines options exigent que le fichier soit journalisé (les options de type *NOACTION où le contrôle d'intégrité se fait au moment de la validation de la transaction (COMMIT).

Règles en modification

Il est toujours interdit de modifier la clé primaire si elle est présente dans un ou plusieurs enregistrements du fichier dépendant (on ne supprime pas un client qui a des factures).

***NOACTION** le contrôle est fait lors du COMMIT (journalisation obligatoire)
***RESTRICT** le contrôle est immédiat

Règles en suppression (DLTRULE)

En suppression d'autres règles sont possibles, la journalisation est nécessaire pour tous les modes à l'exception de *RESTRICT.

***NOACTION** suppression interdite, le contrôle est fait lors du COMMIT (journalisation obligatoire)
***RESTRICT** suppression interdite, le contrôle est immédiat
***CASCADE** entraîne la suppression des enregistrements dépendants,
***SETNULL** entraîne la mise à blanc (NULL) des clés étrangères,
***SETDFT** entraîne la modification des clés étrangères à leur valeur par défaut.

Pour plus d'informations sur ces options, consulter l'aide en ligne de la commande ADDPFCST avec un type *REFCST.

Onglet 'Journal'

Saisir dans cet onglet le nom et la bibliothèque où se trouve le journal. Le journal doit exister et avoir un récepteur.

Voir aussi : gestion des transaction et l'option JOURNALED des infos étendues.

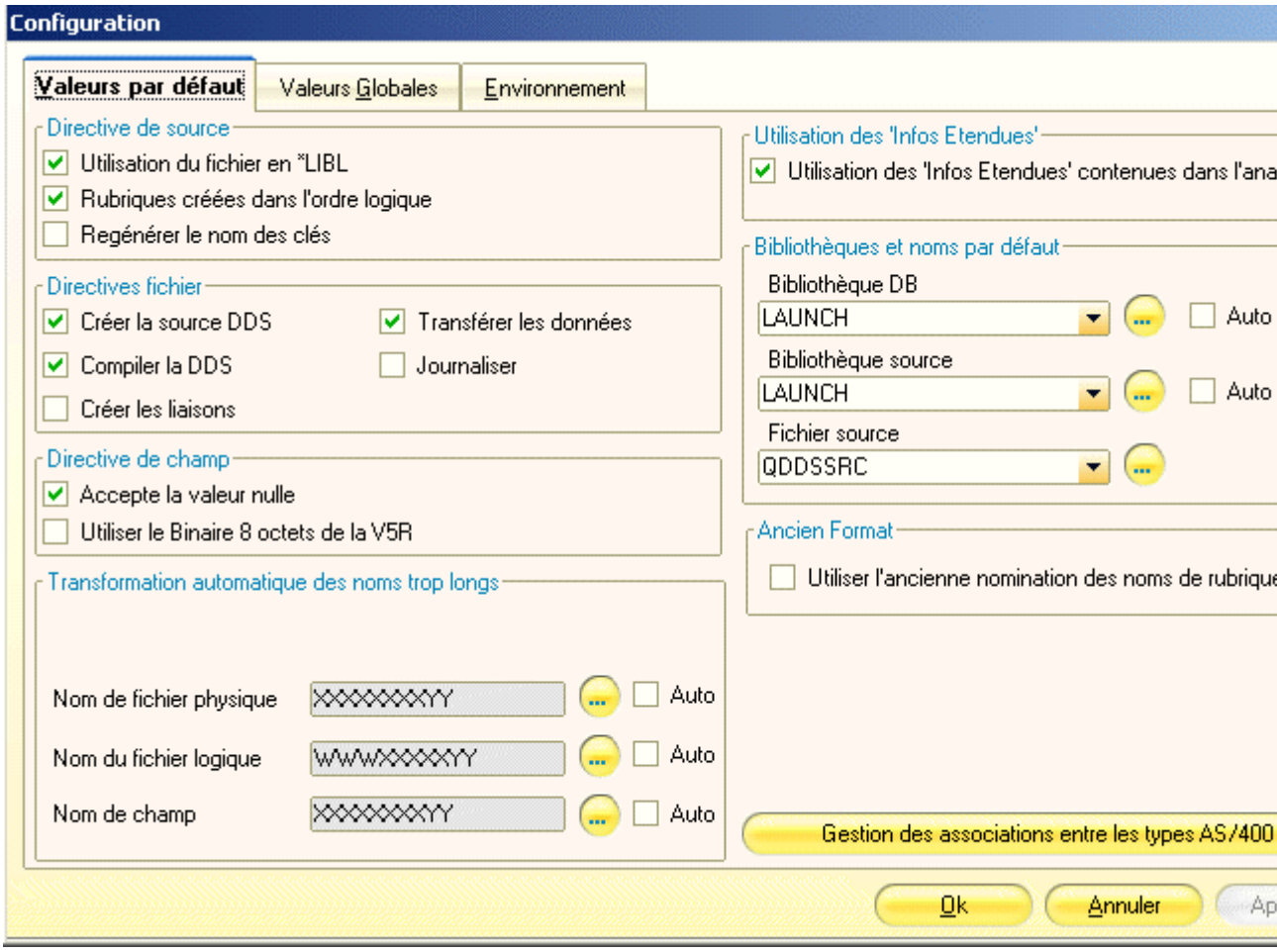
Paramétrages

Dans le menu 'Outils', sélectionnez l'option 'Paramétrages' ou Ctrl - P.

Cette fenêtre permet de sélectionner automatiquement les propriétés communes à tous les fichiers lors du chargement de l'analyse.

La fenêtre de paramétrage se décompose en 3 parties sous 3 onglets :

Valeurs par défaut



Directive de source

*Utilisation du fichier en *LIBL*

Cet option crée des descriptions de fichiers logiques où le paramètre PFILE ne contient pas le nom de la bibliothèque du fichier physique. Le fichier est alors compilé depuis la bibliothèque (par un CHGCURLIB).

Rubriques créées dans l'ordre logique

Si cette option est cochée, les rubriques sont affichées dans l'ordre logique. L'ordre logique correspond à l'ordre d'affichage des rubriques dans les différents éditeurs. Par exemple, l'ordre logique est utilisé pour la création d'une table fichier.

Si cette option est décochée, les rubriques sont affichées dans l'ordre physique. L'ordre physique correspond à l'ordre de création des rubriques dans le fichier de données (fichier ".FIC").

Régénérer le nom des clés

Si cette option est cochée, le nom d'un fichier logique associé à une clé sera automatiquement déterminé par la chaîne de transformation des logiques.

Si cette option est décochée :

- si l'option "Utilisation des infos étendues" est active, le nom du fichier logique est récupéré dans les infos étendues (entrée LF),

- si le nom de la clé est unique et de taille conforme à la limite (10 par défaut) il est utilisé, sinon la chaîne de transformation lui est appliquée.

Directive Fichier

Créer la source DDS

Permet de créer le fichier source (QDDSSRC par défaut), ce fichier source est indispensable à la création des fichiers.

Compiler la DDS

Permet de compiler les DDS et donc de créer les fichiers.

Transférer les données

Permet de copier les données du fichier Hyper File sur l'AS/400.

En cochant cette option, une boite de dialogue demande le répertoire où se trouvent les fichiers Hyper File à copier.

Si des données existent déjà dans le fichier AS/400, les données du fichier Hyper File seront ajoutées.

Créer les liaisons

Permet de créer les contraintes référentielles entre les fichiers sur l'AS/400 à partir des liaisons simples (0,n à 1,1) de l'analyse.

Les liaisons 0,n à 0,1 peuvent être converties en contraintes 0,n à 1,1.

Les autres liaisons ne sont pas des contraintes et ne sont pas exportées.

Journaliser

Permet de journaliser les fichiers, la journalisation est notamment indispensable pour gérer les transactions.

Si le récepteur et le journal n'existent pas, il faut les créer au préalable (CRTJRNRCV puis CRTJRN).

Le constructeur de DDS va ajouter le fichier concerné au journal et démarrer la journalisation (STRJRNPF).

Directive de champ

Accepte la valeur nulle

Si cette option est cochée, toutes les rubriques de l'analyse acceptent la valeur nulle par défaut.

Si cette option est décochée, c'est l'information stockée dans l'analyse WinDev qui sera prise en compte.

Il est possible de préciser pour chaque rubrique si elle accepte ou pas la valeur nulle.

Utiliser le Binaire 8 octets de la V5R2

Utilise le Binaire 8 octets pour les version de l'OS/400 V5R2 et supérieures.

Les numériques de grande taille (entier sur 8 octets - format de 9 999 999 999 999 999 999) seront associés à des binaires 8.

Transformation automatique des noms longs

Voir noms de fichiers et chaînes de transformation.

Nom de fichier physique : permet de fixer la chaîne de transformation des fichiers physiques.

Nom de fichier logique : permet de fixer la chaîne de transformation des fichiers logiques.

Nom de rubrique : permet de fixer la chaîne de transformation des rubriques.

Utilisation des 'infos étendues'

Permet de spécifier l'utilisation des informations étendues des fichiers AS/400 de l'analyse.

Les noms de fichiers (physiques et logiques) et les options (journal) seront récupérés depuis les infos étendues, cette option n'a de sens que si l'analyse ouverte contient des fichiers de type AS/400 avec des infos étendues présentes.

Bibliothèques et noms par défaut

Bibliothèque par défaut : bibliothèque par défaut pour tous les fichiers lors du chargement de l'analyse,

Bibliothèque source : bibliothèque par défaut pour les fichiers sources lors du chargement de l'analyse.

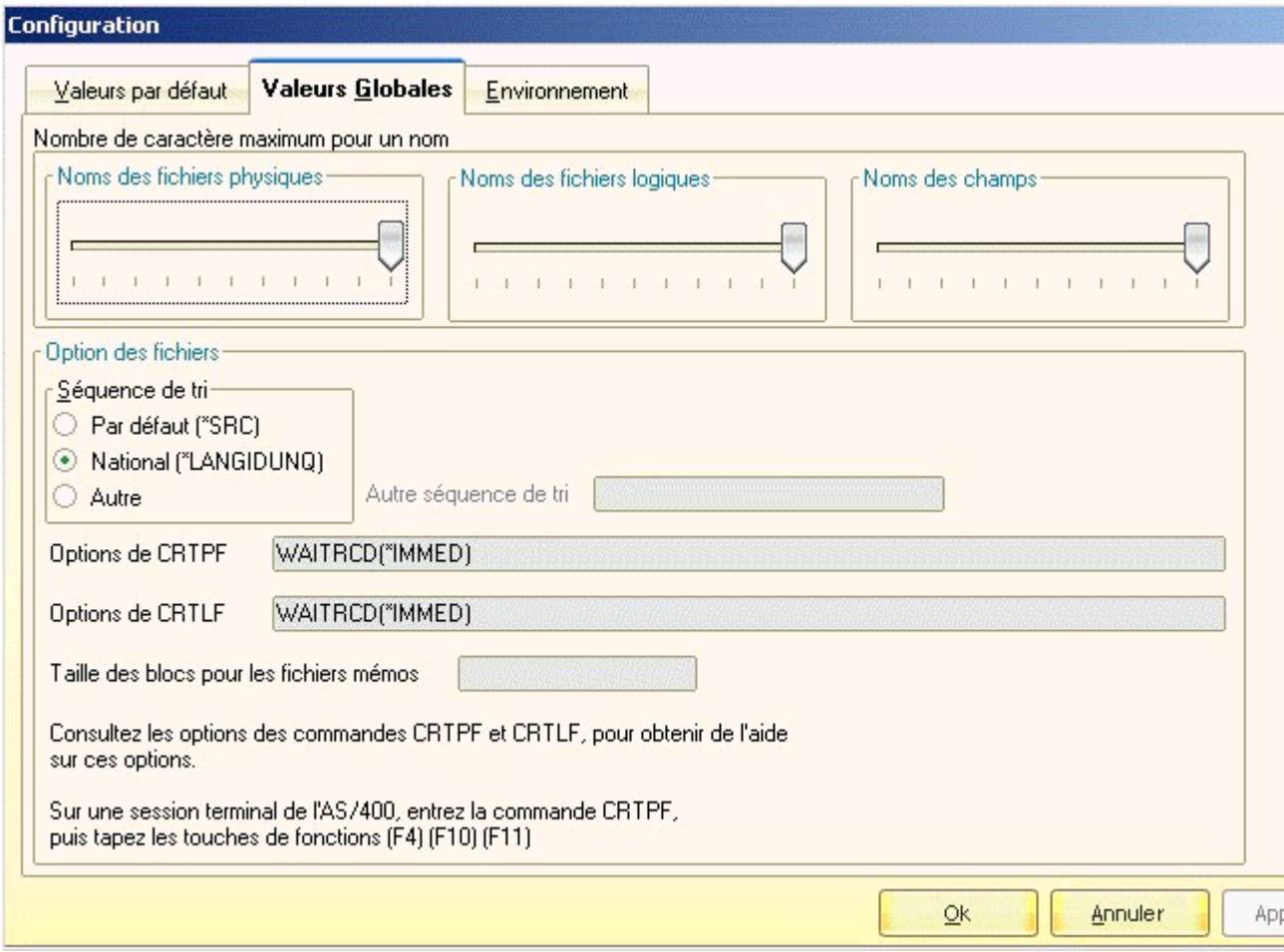
Le sélecteur 'Auto' permet de désactiver l'affichage de la boîte de dialogue qui demande si la modification doit être appliquée à tous les fichiers.

Fichier source : fichier source par défaut lors du chargement de l'analyse (la bibliothèque du fichier source par défaut est la même que la bibliothèque par défaut des fichiers de données).

Ancien format

Utiliser l'ancienne nomination des noms de rubrique tableau.

Valeurs globales



Nombre de caractères maximum pour un nom

Par défaut le nombre maximum de caractères pour le nom d'un fichier, d'un logique et d'un champ est configuré à 10 caractères. Vous pouvez cependant, modifier ce paramètre avec des valeurs comprises entre 3 et 10 caractères.

Il faut dans ce cas avoir des chaînes de transformation qui respectent la nouvelle taille.

Séquence de tri

Chaque fichier est compilé avec une option qui spécifie la séquence de tri

*SRC

*LANGIDUNQ (par défaut)

autre : saisir la séquence de tri

Option des fichiers : cette zone de saisie permet d'ajouter des paramètres de compilation supplémentaires (pour plus d'informations sur les options possibles, voir l'aide en ligne des commandes CRTPF et CRTLF).

Par défaut l'option proposée est WAITRCD(*IMMED), elle permet une réponse immédiate lors d'une tentative de lecture bloquante d'un enregistrement déjà verrouillé.

Mémoriser

Identifiant de connexion

Mot de passe

Environnement

Connexion au démarrage

Permet de sélectionner la connexion définie par défaut lors de l'ouverture du constructeur DDS.

Génération directe

Lorsque le travail de préparation du transfert est terminé, ce qui peut être fait Hors Connexion, vous pouvez lancer la création des fichiers par le menu Génération - Exécuter la création des fichiers sélectionnés.

En fonction des directives et des éléments sélectionnés, le constructeur se connecte alors à l'AS/400 pour lancer la série de commandes. Les éventuelles erreurs sont affichées en cours de traitement et consignées dans un journal qui peut être consulté à la fin.

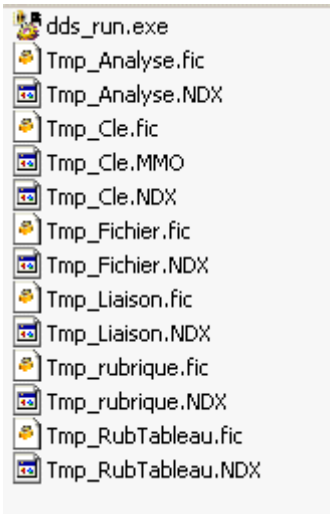
Le profil doit avoir les droits pour créer le source et le compiler.

Génération du Script

Le script est une solution de déploiement souple vers un autre AS/400 et depuis n'importe quel poste client.

Le principe est de créer un petit programme (dds_run.exe) avec toutes les descriptions et les options dans des fichiers Hyper File temporaires (menu Génération - Générer le script de création des DDS).

Dans le répertoire choisi on retrouvera les fichiers suivants :



Il suffit donc de copier ce répertoire vers le poste client sans avoir à y installer la version Développeur et sans copier l'intégralité du répertoire ExportAS400 (ce qui demeure une solution alternative). Le constructeur de DDS peut s'utiliser avec une licence déploiement.

Attention : c'est un programme WinDev 15 qui nécessite donc les dlls WinDev (WD100xxx.dll) et Easycom sur le poste.

Synchronisation

Aucune modification n'est faite sur l'analyse d'origine.
Importation ou synchronisation sont indispensables à la mise à jour des infos étendues.

Les fichiers sont tous de type Hyper File

Le plus simple est d'importer les DDS dans la même analyse ou dans une nouvelle analyse.

L'importation va modifier le type de chaque fichier (Accès Natif AS/400) et associer les infos étendues (aux fichiers, aux clés et aux rubriques).

De manière ponctuelle, il est également possible de transformer le type du fichier (Hyper File Classic en Accès Natif AS/400) en lui associant une connexion (définie dans l'analyse). Il faut dans ce cas faire une synchronisation (depuis l'éditeur d'analyse de WinDev, menu Structure de fichiers - Synchroniser l'analyse avec les bases de données externes) pour récupérer les infos étendues ; associé chaque fichier principal à son fichier physique, chaque clé à son fichier logique et les options éventuelles.

Les fichiers sont de type AS/400

C'est notamment le cas si vous faites des modifications à des fichiers de type AS/400 depuis l'éditeur d'analyse de WinDev. Dans ce cas le constructeur ne va pas créer mais mettre à jour les fichiers sur l'AS/400.

Certaines modifications peuvent être faites dans l'analyse sans devoir exporter à nouveau les DDS (un changement de nom ou de libellé) mais toute modification de la structure (changement du nombre de champs, d'une clé composée...) nécessite une mise à jour du fichier (exportation puis importation ou synchronisation).

Premier point : utiliser l'option "Utilisation des infos étendues" (Paramétrage - Valeurs par défaut) avant l'ouverture de l'analyse afin de récupérer les liens (fichier physique associé au fichier principal, fichier logique associé à chaque clé...) et les options des fichiers (journalisation...).

Après le transfert il n'est pas nécessaire d'importer des fichiers (puisque déjà présents) mais il faut impérativement faire une synchronisation (depuis l'éditeur d'analyse de WinDev, menu Structure de fichiers - Synchroniser l'analyse avec les bases de données externes) pour récupérer les éventuels changements dans les infos étendues.

Evidemment il peut y avoir des risques en cas de suppression ou de modification de rubriques sur l'intégrité des données.

Migration d'une version antérieure

Migration depuis Windev 5.5

Rappel : la migration d'un projet en WinDev 5.5 passe par la version intermédiaire 7.5 (voir documentation WinDev).

Les modifications au niveau d'Easycom concernent :

- les fichiers d'alias dont les informations se retrouvent dans les infos étendues,
- des fonctions obsolètes à remplacer,
- la gestion du retour des fonctions,

Fonctions valides

ASUtilisateur,
ASPropriete.
ASExec,
ASLanceRPC,
ASAppelRTV,
ASRésultatRTV

Nouvelles fonctions

AsErreurAide
ASErreurDonnee

Fonctions obsolètes

ASErreur	intégré dans `HErreurInfo`, étendu par `ASErreurAide` et `ASErreurDonnee`
ASConnecte	intégré dans `HOuvreConnexion`
ASDeconnecte	intégré dans `HFermeConnexion`
ASSource, **ASLitAlias** et **ASEcritAlias**	infos étendues du fichier
ASSQLImmed	intégré dans `HExecuteRequete`

Retour des fonctions

Toutes les fonctions retournent des booléens (Vrai/Faux) à l'exception de ASErreurAide et ASErreurDonnee qui renvoient une chaîne.

Clé réduites

En WinDev 55, pour paramétrer une clé réduite, il fallait procéder comme suit :

Dans le fichier d'alias correspondant (fichier "._as"), à la ligne où figure le nom de la clé, sous la forme "<NomCle> =", ajouter une virgule suivie du nombre de champs clé utiles. On obtenait ainsi :

```
<NomCle> = <bibliothèqueAs>/<FichierAs>, x
```

Il suffit maintenant de modifier directement la description de la clé dans l'analyse en supprimant les dernières rubriques (et uniquement les dernières) ou simplement n'utiliser que les n champs utiles dans la valeur de la clé composée (voir filtre sur une clé composée).

Fichiers d'alias

Les fichiers d'alias étaient utilisés sur les anciennes versions (WinDev 5.5), il s'agit de fichiers textes avec l'extension ._as qui se trouvent dans le même répertoire que le fichier HyperFile (.fic) et qui signalent à WinDev qu'il s'agit de fichiers AS400.

Par compatibilité, il est toujours possible d'utiliser les fichiers d'alias, soit en supprimant les infos étendues du fichier, soit en utilisant la fonction ASPropriete avec l'option ONLYALIAS.

Les fichiers d'alias doivent se situer dans le répertoire courant des fichiers, si ce n'est pas le cas, utilisez la fonction ASPropriete avec l'option ALIASPATH.

Voir : ASPropriete

Migration de projets Windev 7.5 / 8 / 9

Il suffit normalement d'installer la version 2026 d'Easycom pour WinDev et de recompiler le projet.

Si vous maintenez plusieurs applications de versions différentes il est indispensable de conserver les dlls Easycom de chaque version (eac750as.dll, eac800.dll, eac900as.dll, eac1000as.dll) dans le répertoire de l'application ou accessibles par le path.

Les dlls système (easyco32.dll et eac32slf.dll) qui peuvent se trouver dans le répertoire de Windows doivent être mises à jour et sont compatibles avec les versions antérieures.

La mise à jour de la version serveur est nécessaire, elle est également compatible avec toutes les versions antérieures.

Programmes et DataQueues

Introduction

Le principe est d'associer chaque programme et chaque Data Queue à une description de fichier qui contient les paramètres, leur type et si le paramètre est en entrée (IN), en sortie (OUT) ou en entrée/sortie (IN/OUT).

Pour décrire les paramètres, leur type et leur nature (entrée / sortie), vous disposez d'un outil spécifique, le "Constructeur RPC/DTAQ". Toutes les descriptions sont stockées sur l'AS400, dans les fichiers YPROCHDR et YPROCPARMS.

Pour importer ces descriptions, saisir *PGM (programmes) ou *DTAQ (data queue) dans la zone "Bibliothèque" ou les sélectionner dans la liste des fichiers.

L'appel d'un programme avec paramètres ou l'ajout dans une data queue se fait par un opération d'écriture sur le fichier image par l'appel à la fonction ASLanceRPC .

La récupération des paramètres en retour d'un programme ou la lecture de la data queue se font par une opération de lecture par clé sur le fichier image.

Voir exemples ASLanceRPC et Data Queue.

Ces descriptions peuvent être copiées ou transférées d'un AS/400 vers un autre.

Décrire les programmes natifs AS/400

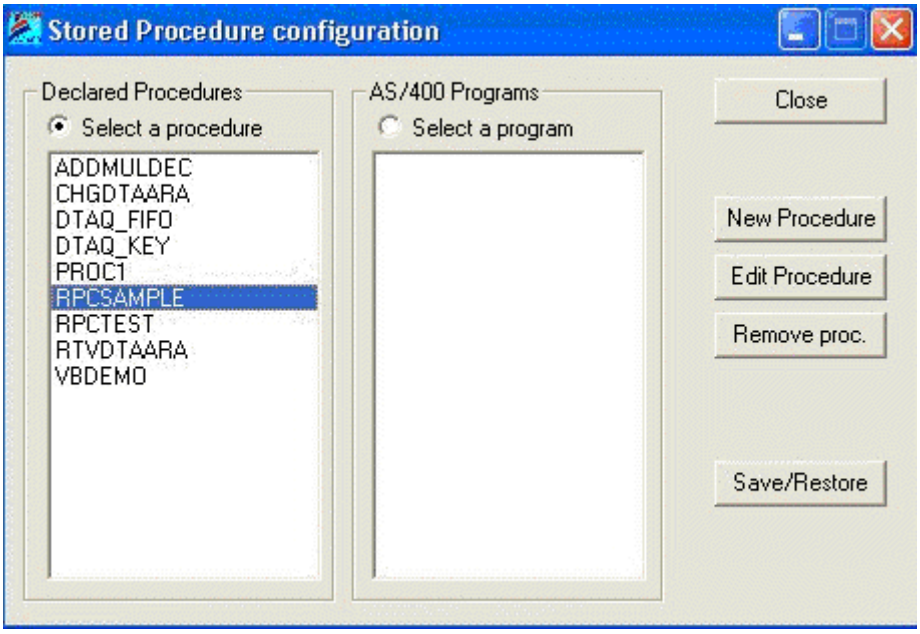
Voir : Importation de programmes ou de Data Queue

EASYCOM permet d'appeler les programmes natifs de l'AS/400, CL, RPG ou encore des procédures stockées.

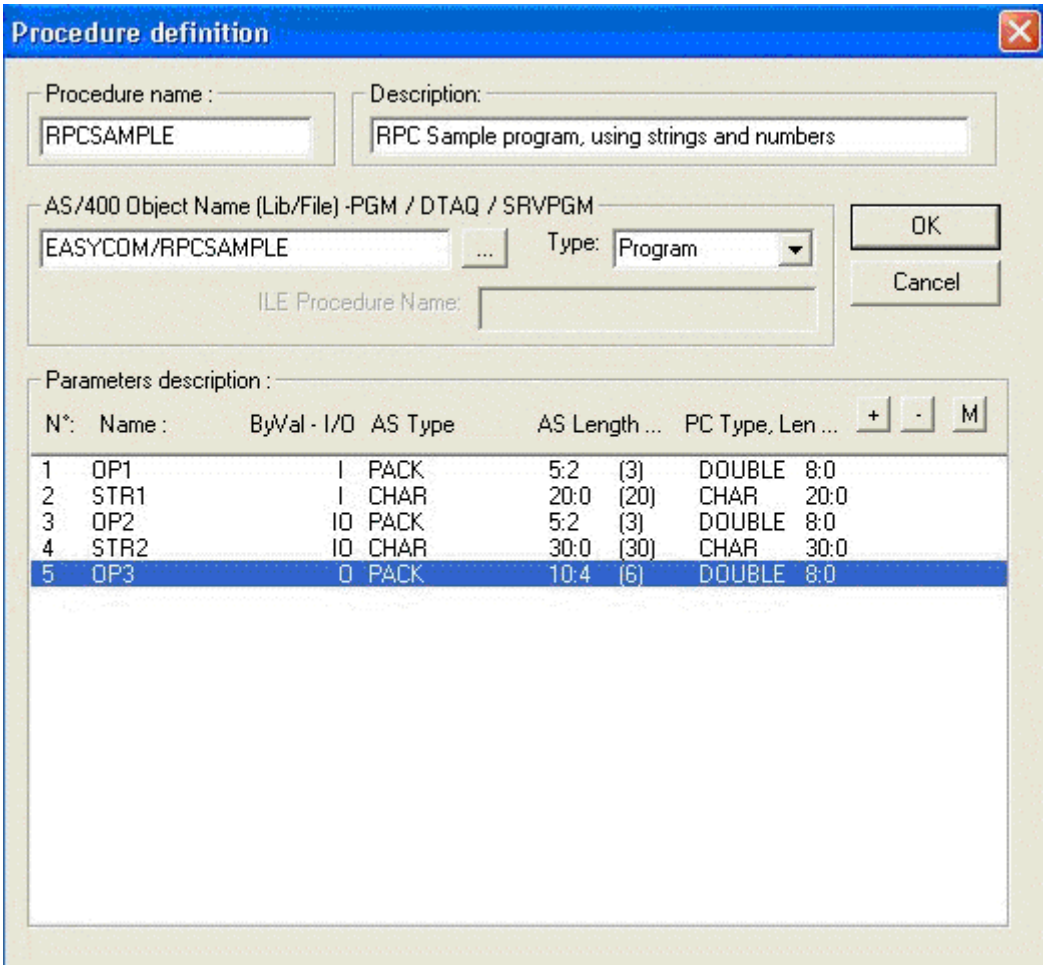
Pour cela EASYCOM a besoin de **la description de ces programmes** qu'il stocke sur l'AS/400, dans les fichiers YPROCHDR et YPROCPARMS de la bibliothèque EASYCOM.

La description des programmes et des data queues se fait par le constructeur DTAQ-RPC. Le principe de base consiste à préciser l'ensemble des paramètres, leurs types et leur utilisation (entrée, sortie, entrée-sortie) pour l'appel du programme.

Le premier écran présente les procédures existantes (stockées sur l'AS/400) et permet de créer, modifier ou supprimer ces dernières. On peut également sauvegarder dans un fichier au format texte et côté PC ces descriptions pour les transférer ensuite sur un autre AS/400.



Un nouveau nom est donné à la procédure, et n'a pas besoin d'être identique au programme qui lui sera associé.
Un programme natif AS/400 (CL, RPG, COBOL, C, …) est associé à la procédure.
La bibliothèque peut être omise, ou remplacée par *LIBL.
La description est un texte libre qui sera visible lors du parcours des procédures par les postes clients.
Chaque paramètre d'appel du programme est décrit, afin de donner son type et sa taille.



Chaque paramètre peut être considéré comme un champ dans une table de base de données.

Il a donc un nom, auquel l'application pourra faire référence.

Des paramètres destinés à envoyer des valeurs au programme appelé sont considérés comme paramètres en entrée (**IN**).

Les paramètres destinés à recevoir une valeur au retour de l'appel sont considérés comme paramètres en sortie (**OUT**).

Des paramètres modifiés par le programme sont en Entrée et Sortie (**IN /OUT**).

Par défaut, tous les paramètres d'un programme de l'AS/400 sont en Entrée et Sortie.

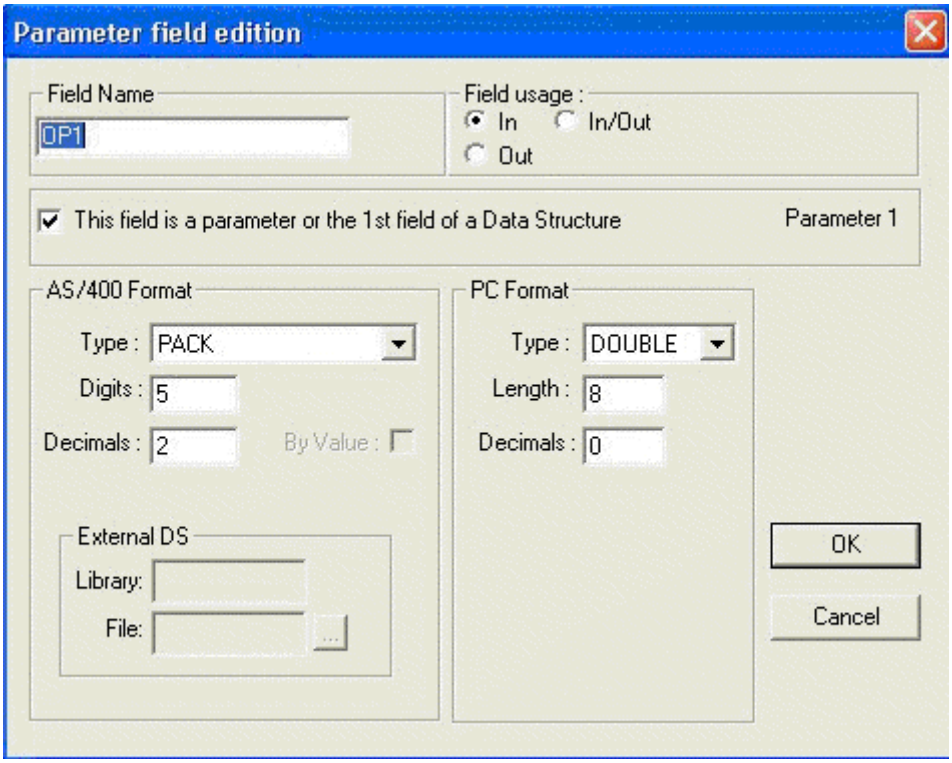
C'est la logique du programme qui peut changer cette propriété.

Si un paramètre d'appel du programme est une structure (DS : Data Structure), chaque champ de la DS doit être décrit individuellement.
Pour le premier champ uniquement, la case à cocher est : "This field is a parameter or the 1st Field of a Data Structure."

Le type du paramètre attendu par le programme sur l'AS/400 doit être donné avec exactitude :

CHAR :	Donnée de type caractère
BIN2 :	Donnée numérique entière sur 16 bits
BIN4 :	Donnée numérique entière sur 32 bits
PACK :	Donnée numérique condensée (DECIMAL)
	C'est dans ce format que les données numériques sont gérées par le CL (type CL *DEC)
ZONED :	Donnée numérique étendue (NUMERIC)
DATE :	Date AS/400 sous la forme jj-mm-aaaa

TIME :	Heure dans le format hh :mm :ss
FLOAT :	Valeur numérique en flottant simple précision
DOUBLE :	Valeur numérique en flottant double précision
TIMESTP :	Champ horodatage (timestamp)
GRAPHIC :	Donnée de type caractère, ne devant pas être convertie
EXTERNAL DS :	La structure est décrite par une Data Structure externe, c'est-à-dire un fichier physique



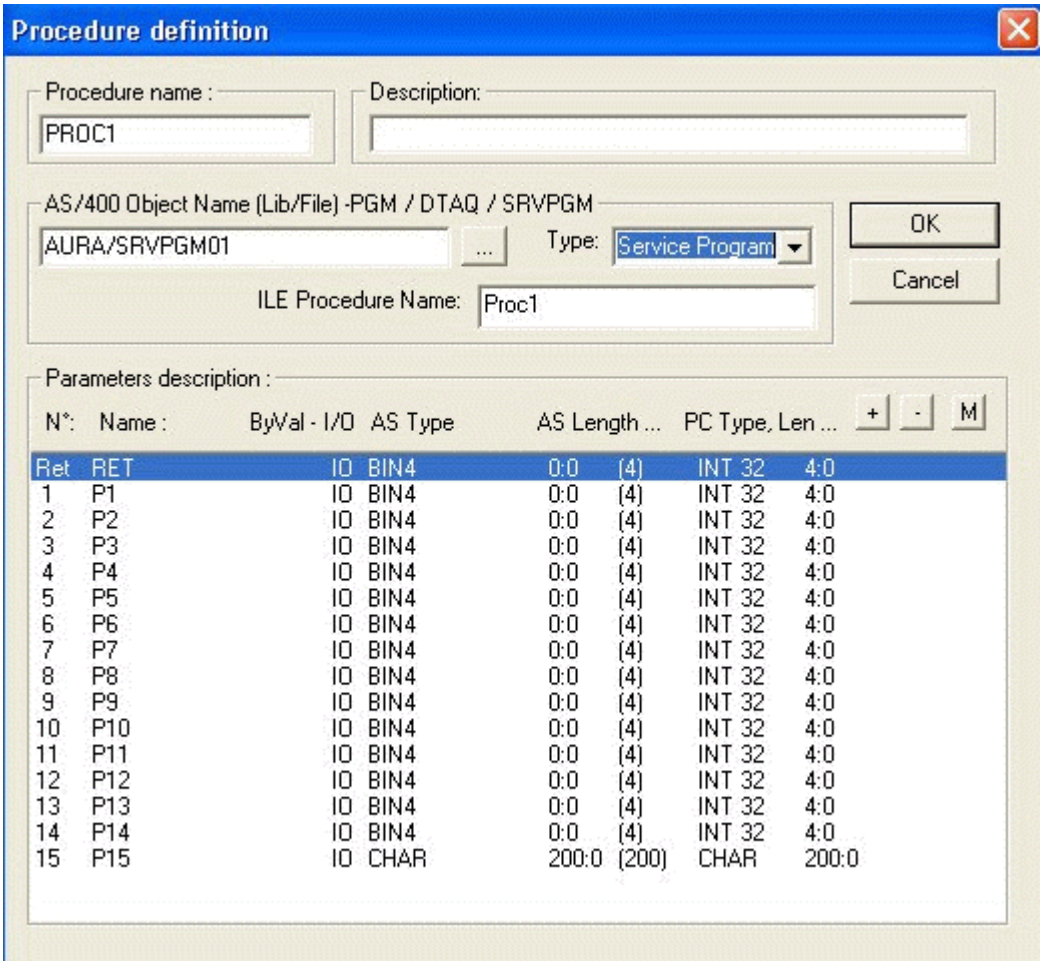
Les programmes décrits pourront ensuite être "ouverts" comme des fichiers, appelés en simulant une écriture et les valeurs en sortie pourront être lues en simulant une lecture.

Appel de procédure de programme de service

La configuration d'une procédure de programme de service se fait comme pour un programme OPM, de type *PGM.

L'objet désigné dans le champ « AS/400 Object Name » doit être de type *SRVPGM.

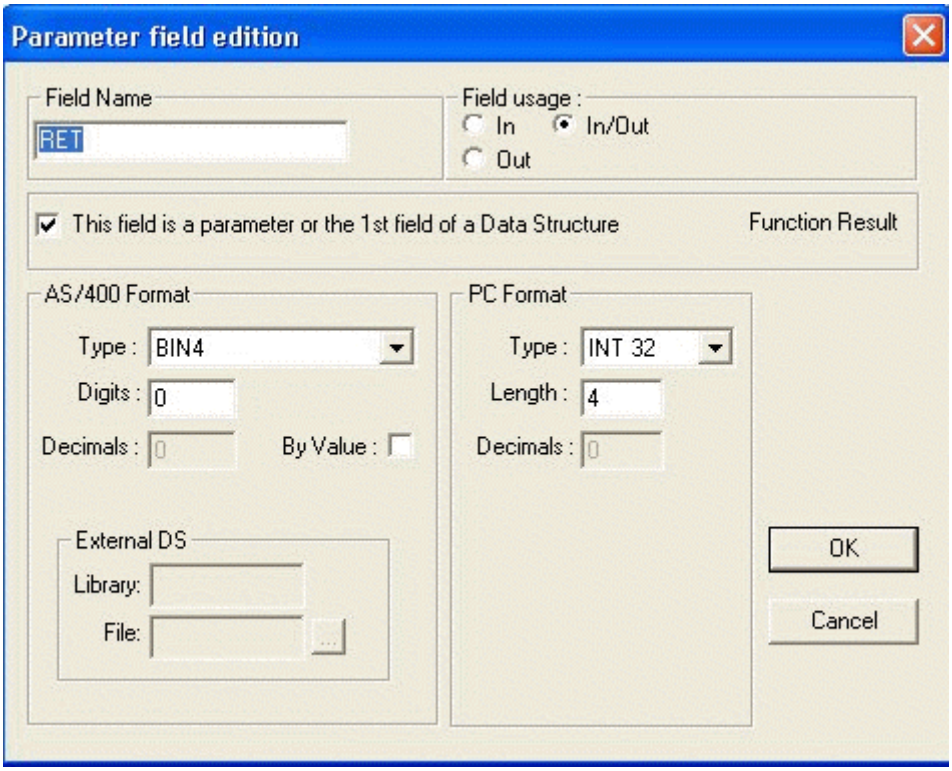
Le type doit être « Service Program ».



Le premier paramètre décrit correspond au retour de la procédure.

Seul les retours de type « Entiers 32 bits » sont acceptés.

Les paramètres sont ensuite décrits, comme pour un programme OPM avec 16 paramètres maximum, plus la variable de retour.



Pour chaque paramètre, une option a été ajoutée : « By Value ».

Cette option n'est valable que pour les paramètres de type « Entier 32 Bits », c'est à dire « BIN4 ».

Si elle est cochée, cette option indique que la procédure attend ce paramètre par valeur, et non par adresse.

Importation de programmes ou de Data Queue

Lors de l'import de fichier, Il vous suffit, depuis l'utilitaire d'utiliser et de spécifier les valeurs spécifiques "***PGM**" ou "***DTAQ**' comme nom de bibliothèque.

Procédez au reste de l'importation comme vous le faites pour les autres types de fichiers

Remarques :

Si votre Data Queue est de type FIFO, la lecture se fait sur une clé simple sur la première rubrique (timeout).

Si votre Data Queue est de type KEYED, la lecture se fait sur une clé composée sur les quatre premiers champs.

NB : Pour les DTAQ, 3 paramètres sont obligatoires.

Migration des procédures et data queues d'un AS/400 vers un autre

Lorsqu'un développeur travaille sur un AS/400 et qu'il crée des procédures et des data queues qui devront par la suite être utilisées sur un autre AS/400, il est nécessaire de transférer les descriptions de ces procédures et data queues sur le nouvel AS/400.

Trois fichiers stockent les descriptions : YPROCHDR, YPROCPARMS et YPROCPGM.

Sur l'AS/400 elles sont stockées par défaut dans la bibliothèque EASYCOM mais il est possible de les mettre dans une autre bibliothèque auquel cas elles seront recherchées dans la LIBLE du profil connecté.

Si les deux AS/400 sont connectés ont peut évidemment transférer directement de l'un vers l'autre les trois fichiers ci-dessus.

Autrement le constructeur DTAQ-RPC propose une option d'importation/exportation des descriptions dans un fichier texte, on peut donc sauvegarder depuis le poste du développeur les descriptions nécessaires pour les restaurer sur un poste du client final.

Utilisation DATA QUEUE

Les DataQueues sont des objets permettant des canaux de communication entre programmes. Elles sont plus puissantes et plus simples d'emploi que les DataAreas . Tout comme elles, elles permettent l'échange de données entre programmes de type différent (JAVA avec CL par exemple) et bien sûr les applications Easycom.

On distingue :

les DTAQ FIFO (First In, First Out) : réception de msg1 puis msg2 ,3,4...

les DTAQ LIFO : réception de msg4 puis msg3,2,1...
et les KEYED DTAQ qui permettent d'accéder à un message donné suivant une clé .

La Data Queue doit avoir été décrite (voir constructeur) et importée dans l'analyse.

Une Data Queue est écrite de manière directe, mais la lecture peut se faire en mode direct ou temporisée.

Ecriture

Quel que soit le type d'une Data Queue, une écriture est réalisée par la commande HAjoute().

Note: Il n'est pas utile de mettre des valeurs dans les champs de contrôle (les 3 ou 4 premiers, en fonction du type de la Data Queue).

Exemple :

```
// Initialise les rubriques...
DTAQ_FIFO.Data = DATA

// Ecriture dans la DataQueue
HAjoute(DTAQ_FIFO)
```

Lecture directe

La récupération des messages est immédiate et se fait avec la commande HLitRecherchePremier.

```
// Lecture de la DataQueue
HLitRecherchePremier(DTAQ_FIFO,Timeout,"000000")

// Affichage du résultat
SI H.EnDehors = Vrai ALORS
 Info("DataQueue vide !")
SINON
 // Affichage...
 SAI_MSG1=DTAQ_FIFO.Data
FIN
```

Lecture temporisée

Si : la Data Queue contient au moins un message, la récupération est immédiate, sinon une temporisation paramétrable (en secondes) est lancée pour attendre un nouveau message.

Ceci est réalisé par la commande HLitRecherchePremier, en indiquant le délai dans la clé simple "TIMEOUT".

```
txtAttente est une chaîne

// Lecture de la Data Queue
txtAttente = TIMEOUT
HLitRecherchePremier (DTAQ_FIFO, TIMEOUT, txtAttente)

// Affichage du résultat
SI HEnDehors ALORS
    Info("DataQueue vide !")
SINON
  DATA = DT.DATA
FIN
```

Attention : dans ce cas l'application reste "bloquée" sur la fonction pendant toute la durée du timeout à moins que la Data Queue ne reçoive un message.

Data Queue de type KEYED

Il n'existe qu'une seule façon de lire une Data Queue KEYED : lecture par clé par la commande HLitRecherchePremier.

La composition de la clé suit les règles suivantes:

Le champ "TIMEOUT" indique en secondes une valeur d'attente avant un nouvel essai (il est possible de ne rien indiquer), dans ce cas la réponse est immédiate.

Le champ "FILER" doit être vide,

Le champ "ORDER" donne le type de recherche :

EQ	=	égal
NE	<>	différent
GT	>=	strictement supérieur
GE	>	supérieur ou égal
LT	<	strictement inférieur
LE	<=	inférieur ou égal

Le 4ème champ contient la valeur de recherche, il dépend de votre Data Queue.

Exemple

```
txtCle est une chaîne

// Lecture de la DataQueue

txtCle=HConstruitValClé(

HLitRecherche(DTAQ_KEY, DTAQ_KEY, txtCle)

// Affichage du résultat
SI H.EnDehors = Vrai ALORS
```

```
 Info("DataQueue vide !")
SINON
   // Affichage...
  DATA = DT.DATA
FIN
```

Utilisation DATA AREA

Pour accéder aux Data Area il suffit d'envoyer les commandes avec les paramètres avec la fonction AsExec. pour l'écriture et d'utiliser les fonctions ASAppelRTV et ASRésultatRTV pour la lecture.

Exemple

```
eacret is int
CmdLine is string
RetVal is string

// Connect to AS/400
IF NOT
HOpenConnection("MyConnection",user,pwd,ipadress,hNativeAccessAS400,hOReadWrit
e,"") THEN
 Info(HErrorInfo())
END

// Write to DATAARA
CmdLine="chgdtaara dtaara(easycom/dtaara) value('Hello World')"
IF NOT ASExec(CmdLine) THEN
 Info(ErrorInfo())
END

// Retreive DTAARA value
CmdLine = "rtvdtaara dtaara(easycom/dtaara *ALL) rtnvar(&var1)"
eacret = ASRtvCall(CmdLine)
IF NOT eacret THEN
 Info(ErrorInfo())
END

// Lecture du Result
RetVal = ASRtvResult("var1")
IF NOT RetVal = "0" THEN
 Info("New DTAARA value is : " + RetVal)
ELSE
 Info("L'appel a échoué")
END
HCloseConnection("MyConnection")
```

WebDev

Déploiement WebDev

Sur un serveur WebDev, les boites de dialogues Easycom et WinDev sont inactives. Il faut donc gérer la connexion et toutes les erreurs de manière plus rigoureuse.

Déploiement

Sur un poste de développement, où Webdev 2026 et l'accès natif AS400 sont installés, vous devriez trouver les DLL EASYCOM (eac3100as*.*) ici (par défaut) :

C:\PC SOFT\WINDEV Suite 2026\Programmes\Framework\Win32x86 pour les DLL 32 bits.
C:\PC SOFT\WINDEV Suite 2026\Programmes\Framework\Win64x86 pour les DLL 64 bits.

Copiez les DLL, selon votre cas, 32 et/ou 64 bits, sur votre serveur Web : dans C:\Webdev 2026, à côté des autres DLL Windev/Webdev.

Prestarts jobs

Afin d'accélérer les ouvertures de sessions (de connexion), la configuration et l'utilisation des Prestarts jobs est recommandée.

Verrouillages

Sous WinDev, un `HModifie` sur un enregistrement préalablement ouvert en lecture seule va déclencher un mécanisme de vérification entre les données en cache et les données rechargées, c'est à dire qu'une nouvelle lecture bloquante est faite.

Sous WebDev il faut être plus rigoureux et ne jamais ouvrir en lecture seule un enregistrement susceptible d'être modifié ou bien gérer les éventuelles erreurs avec la fonction `HSurErreur(hErrBlocage)`.

Voir : WebDev : spécificités accès natif (verrouillages)

L'option DRVOPTIMISTIC des infos étendues de la connexion peut également être utilisée.

Easycom serveur

Easycom serveur

Easycom serveur est la partie AS/400 commune à tous les produits Easycom.

Il s'agit du cœur des produits middleware tels qu'Easycom For PHP, Easycom For WinDev, Easycom For Delphi, .NET, etc.

Easycom serveur consiste en une bibliothèque sur le system i avec un sous-système. La partie client se connecte à un travail du sous-système qui le relie à un travail propre à la connexion (soumis ou bien à démarrage anticipé).

La partie client peut être sur Windows, PASE, AIX, Linux, et bien d'autres plateformes.

La technologie est la propriété de la société Aura Equipements, France.

Cette documentation présente comment installer et configurer Easycom sur la partie client et serveur.

Installation et Configuration d'Easycom Serveur

Installation du serveur Easycom

Voir Installation du serveur Easycom sur l'AS400.

Configuration du serveur Easycom

Voir Configuration et administration du serveur EASYCOM .

Caractéristiques Easycom

Fichiers de configuration de EASYCOM serveur

Sur l'AS/400, l'ensemble des objets du serveur EASYCOM est dans la bibliothèque **EASYCOM**.

Un seul objet complémentaire est créé au moment du premier lancement du serveur : Objet EASYCOM, type *FILE, librairie QGPL.

La bibliothèque EASYCOM est automatiquement ajoutée dans la liste des bibliothèques en ligne (ADDLIBLE) pour chaque travail démarré pour les postes clients. Il n'est donc pas nécessaire de l'ajouter explicitement dans les JOBD des utilisateurs.

Les objets suivants peuvent subir des modifications après l'installation du serveur sur l'AS/400 :

AURA	*FILE	Modifié au moment de l'enregistrement de la licence d'utilisation.
EACSESSION	*FILE	Modifié au moment de l'enregistrement de la licence d'utilisation.
YPROCHDR	*FILE	Modifié par l'ajout d'une description d'un programme natif AS/400 destiné à être appelé depuis des applications clientes.
YPROCPARMS	*FILE	Modifié au moment de la description des paramètres d'appels des programmes natifs AS/400.
CFGEAC	*DTAARA	Contient les paramètres de la commande CFGEAC
CFGEACSSO	*DTAARA	Contient les paramètres de la commande CFGEACSSO
EASYCOM	*SBSD	Contient les paramètres de la commande CFGEACSBS
EAC_EIM	*USRSPC	Contient les paramètres de la commande CFGEACSSO en mode *EIM

En cas de mise à jour du serveur EASYCOM sur l'AS/400, ces objets sont restaurés vers la nouvelle version tout en conservant leurs valeurs initiales.

Connectivité IPv6

Easycom est entièrement compatible IPv6.

Il n'y a pas de configuration particulière pour autoriser les connexions IPv6. La seule condition est d'utiliser une version minimum pour les programmes EASYCOM et EASYCOMD ainsi que d'OS/400:

- Le programme EASYCOM doit être en version 4.60.10 ou plus
- Le programme EASYCOMD doit être en version 3.0.3 ou plus
- La version d'OS/400 doit être V5R3 ou plus

Pour bénéficier d'une connectivité entièrement IPv6, le client Easycom doit également être compatible IPv6. Il faut donc vérifier la compatibilité de chaque produit client Easycom.

Bien qu'il soit conseillé d'utiliser un nom de machine et non une adresse, il est possible de spécifier une adresse au format IPv6, sous la forme suivante :

2001:db8::1428:57ab

ou bien

[2001:db8::1428:57ab] :6077

6077 étant le port TCP/IP à utiliser.

Il est possible de vérifier si une connexion a été établie en IPv6 à l'aide de la commande NETSTAT de l'OS/400.

Le mode de connexion est également visible dans le fichier trace (généré via CFGEAC ou par l'application cliente).

Remarques:

- L'adresse IP apparaît en syntaxe IPv6 au niveau de tous les programmes d'exit d'EasycomD.
- La version d'IP est disponible avec deux programmes d'exit: EACTCPP01 and EACLOG002
- Les connexion en IPv6 et IPv4 sont indifféremment acceptées par défaut. Il est possible de restreindre à un procole à l'aide du program d'exit EACLOG002.

Les prestart job

Les "prestart" jobs, ou travaux à démarrage anticipé, permettent de gagner du temps à la connexion EASYCOM . Leur utilisation est particulièrement adaptée si des applications se connectent et déconnectent fréquemment (tout particulièrement les applications web).

Avantages:

- Démarrage plus rapide de la connexion (connexion quasi instantanée).
- Possibilité de préparer un environnement avant la connexion (mais sans connaître à l'avance quel va être l'utilisateur)

Inconvénients:

- le nom du job est le même pour toutes les connexions, et ne peut pas être déterminé par le client.
- L'utilisateur effectif n'est visible dans la liste affichée par WRKACTJOB qu'en V5R4 ou plus. Pour connaître l'utilisateur effectif en V5R3, utiliser l'option 5 de WRKACTJOB (gérer), puis choisir l'option 1 (état du travail).

Voici comment configurer, par l'exécution de ces 4 commandes AS/400, les travaux à démarrage anticipé :

(toutes les connexions devront être arrêtées)

```
ENDSBS SBS(EASYCOM) OPTION(*IMMED)

ADDPJE SBSD(EASYCOM/EASYCOM) PGM(EASYCOM/EASYCOM) INLJOBS(10)
JOB(EASYCOMPJ) JOBD(EASYCOM/EACJOBD) CLS(EASYCOM/EACCLS)

CFGEACSBS SBS(EASYCOM) MODE(*PJ) PJSTART(*YES)

STRSBS SBSD(EASYCOM/EASYCOM)
```

Pour revenir en configuration classique, arrêter à nouveau le sous-système, utiliser RMVPJE pour supprimer la définition des PJ, puis utiliser la commande `CFGEACSBS` avec **PJSTART(*NO).**

Prestarts Jobs et Exit Programs

Lorsque les prestarts jobs sont activés, vous pouvez associer un programme qui sera automatiquement exécuté:

- à l'initialisation des Prestarts Jobs (EACPJINI)
- à chaque nouvelle connexion (EACTCPP01, puis EACLOG002 et EACTCP002)

IPL

Le sous-système Easycom doit être en cours d'exécution, ainsi que le travail Easycomd, pour qu'Easycom soit opérationnel. EasycomD démarre automatiquement en même temps que le sous-système.

Il est donc nécessaire de prévoir un démarrage automatique du sous-système lors des IPL.

La méthode la plus utilisée est la modification du programme de démarrage QSTRUP.

QSTRUP est un programme CL. Pour le modifier, il faut procéder en 3 étapes :

- Extraire le source CL par la commande RTVCLSRC QSYS/QSTRUP,

- Modifier le source (ajouter le démarrage d'Easycom par l'ajout de la commande STRSBS EASYCOM/EASYCOM en fin de programme).

- Recompiler le programme CL par la commande CRTCLPGM.

LIBL par défaut

Maintenant, la LIBL du travail EASYCOM est constituée, dans l'ordre :

- Des bibliothèques issues de la JOBD de l'utilisateur,
- Des bibliothèques issues de EACJOBD, si présente la la bibliothèque Easycom,
- De la bibliothèque EASYCOM.

Par défaut, la LIBL de EACJOBD est initialisée à *NONE.

Attention ! Cette description est valable pour EASYCOM Serveur version 4.58.80 et ultérieurs.

CCSID et SRTSEQ par défaut

Les CCSID et STRSEQ (séquence de tri) peuvent être configures à l'aide de la commande CFGEAC.

Le comportement par défaut d' Easycom est d'utiliser le CCSID par défaut du système. Il est possible d'utiliser par exemple la valeur associée au profil utilisateur à l'aide de la commande CFGEAC.

Il est préférable d'utiliser un CCSID approprié pour le travail Easycom, soit identique pour tout le monde, soit configuré par utilisateur. Cela permet aux programmes Easycom d'effectuer les conversions appropriées entre les différents CCSID utilisés dans les fichier (et même Unicode), et la partie cliente Easycom.

Timeout sur appel de programme

Il est possible de configurer par le programme CFGEAC un temps d'exécution maximum (timeout) pour les programmes appelés depuis les applications clientes. (fonction ASLanceRPC).

En l'absence de réponse du programme après le délai configuré, l'appel est annulé (comme si l'on avait demandé un appel système en mode terminal), et la fonction appelante (côté client) va renvoyer une erreur et le programme client peut se poursuivre. Attention, les fichiers et ressources allouées par le programmes pourront dans ce cas rester allouées!

Par exemple un AsLanceRPC attendra la réponse du serveur, si pour une raison quelconque le programme ne répond pas, le job annule l'appel et l'erreur retournée par la fonction permet alors de gérer le problème ; envoyer un message à l'administrateur, fermer et relancer la connexion pour les traitements ultérieurs...

Protection de l'accès à Easycom

Easycom gère la sécurité au niveau utilisateur: un programme EASYCOM peut être utilisé sur un AS/400 une fois l'utilisateur mot de passe validée par le système via Easycom. Les traitements seront exécutés avec l'autorité de l'utilisateur utilisé et toutes les permissions et droits définis sur le système seront respectés.

Toute connexion ouverte requiert une validation complète par utilisateur et mot de passe (ou bien une authentification Kerberos si ce type de connexion est configuré).

L'utilisation d'exit program et éventuellement du mode « protégé » d' Easycom peut améliorer encore la sécurité.

Il existe notamment les programmes d'exit suivants:

- EACP003 : autorise le lancement d'un programme par l'utilisation d'un mot de passe complémentaire, fourni par le programmeur PC et analysé par cet exit program. Cet exit program permet par exemple de ne valider que les programmes « maison », et certains développeurs.
- EACTCP003 et EACTCP002 : contrôle l'adresse ip et station en cours de connexion. Permet également de changer l'utilisateur en effectif ou encore de soumettre manuellement le travail Easycom.
- EACTCPP01 : contrôle de l'appelant, avant l'authentification (par addresse IP, protocole, …)
- EACLOG002 : contrôle les signatures en général (et Kerberos en particulier), après validation par Easycom
- EACSSO001 : contrôle le Single Sign-On 'Easycom' (obsolete, à remplacer par EIM si l'OS/400 est en V5R3 ou plus)
- EACSOPEN, EACSCALL et EACSRMCD : permet de contrôler au niveau ressource (pour autoriser ou interdire des fichier, programmes en fonction de l'utilisateur ou bien générer un historique d'utilisation).

Single Sign On - EIM

Qu'est-ce que EIM?

Le Single Sign On (SSO) en mode EIM est l'implémentation du système de single sign-on d'IBM.

Le principe est basé sur l'utilisation d'un serveur d'authentification (nommé : credential management server) unique, Kerberos. Lorsque l'utilisateur se connecte à sa station, le serveur Kerberos lui fournit un **ticket**.

Lorsqu'une connexion basée sur EIM s'opère, ce **ticket** est fourni à l'IBM i à la place du traditionnel utilisateur/mot de passe. Ce ticket est ensuite validé par le serveur Kerberos depuis l'IBM i, et enfin un utilisateur OS/400 correspondant au nom d'utilisateur du ticket (l'utilisateur Windows) est fourni et utilisé par Easycom.

Donc le mot de passe n'est plus utilisé pour l'accès à l'AS/400, et on peut mettre le mot de passe utilisateur à *NONE pour une meilleure sécurité. L'utilisateur sera obligé d'utiliser une authentification Kerberos pour se connecter.

L'installation de EIM sur IBM i

L'installation d'EIM sur IBM i consiste en les étapes suivantes :

- Créer un domaine dans EIM
- Ajouter le domaine dans la gestion des domaines
- Créer une définition de registre d'utilisateur source dans EIM.
- Créer un identificateur d'utilisateur dans EIM.
- Créer une association cible dans EIM pour l'identificateur d'utilisateur.
- Créer une association source dans EIM pour l'identificateur utilisateur.
- Tester la connexion au contrôleur de domaine EIM
- Configurer la fabrique de connexions de jetons d'identité EIM

Les informations détaillées sur l'installation d'EIM sur IBM i, sont dipsonibles ici :
https://www.ibm.com/docs/fr/was/9.0.5?topic=eim-configuring

Dès que cela fonctionne avec Client Access, reste à configurer Easycom.

EIM avec Easycom

Pour pouvoir utiliser Easycom avec l'EIM, il est nécessaire de réaliser les étapes suivantes :

1. Installer et configurer EIM sur l'IBM i et le contrôleur de domaine
2. Autoriser l'accès du fichier keytab à l'utilisateur QTCP. QTCP est l'utilisateur sous lequel tourne le travail EASYCOMD.

   ```
   CHGAUT
   OBJ('/QIBM/UserData/OS400/NetworkAuthentication/keytab/krb5.keytab')
   USER(QTCP) DTAAUT(*R)
   ```

3. Activer l'authentification Kerberos :

   ```
   CFGEACAUTH LIB(EASYCOM) KERBAUTH(*ON)
   ```

   Remarque : Il est possible d'utiliser l'authentification par certificat, ceux-ci étant référencés dans la base données EIM. Dans ce cas il n'est pas nécessaire d'activer l'authentification Kerberos.

4. Configurer Easycom pour utiliser EIM sur le serveur :

   ```
   CFGEACEIM LIB(EASYCOM) ACTIVE(*YES) EIM_LDAPU(administrator)
   EIM_LDAPPW(xxx)
   ```

   Cela va permettre à Easycom de se connecter au système LDAP afin de pourvoir connaître l'utilisateur AS/400 correspondant au ticket Kerberos.

5. Définir (optionnel) l'exit program « EACLOG002 »
6. Mettre à jour les applications pour utiliser la valeur spéciale « *KERBAUTH » comme login.

L'implémentation de l'EIM côté client est très simple. Il suffit de spécifier « *KERBAUTH » comme nom d'utilisateur, en disposant de versions récentes des DLL clientes.

Il existe des codes d'erreur TCP/IP spéciaux (négatifs), pour les différents types d'erreurs Kerberos (ticket expiré, …), accompagnés de textes d'erreurs (provenant de l'IBM i ou du client).

Un moyen simple de tester la configuration sans modification des programmes : saisir *KERBAUTH comme nom d'utilisateur et laisser le mot de passe à blanc. Vous pourrez ensuite modifier les programmes pour y introduire cette valeur spaciale.

Voir aussi

L'installation de EIM sur System i

EIM Problèmes courants

CFGEACEIM

EIM : Problèmes courants

a. *Les noms de domaine doivent correspondre*

Le nom de domaine qui est configuré au niveau d'iSeries navigator doit correspondre au nom de domaine de la machine.

Dans le cas contraire, l'erreur suivante sera rencontrée au niveau client *"the specified target is not known or inaccessible"* (avec le code d'erreur tcp/ip -14)

Voici comment vérifier :

Etape 1 : pour connaître quel est le nom de domaines de la machine, lancez un invite de commandes sur la **machine cliente** :

Tapez « nslookup », et entrez ensuite le nom de l'IBM i. Par exemple :

```
Default server : domain_controller.domain-name.com
Address: 194.206.160.4

> my_iseries
Server : domain_controller.domain-name.com
Address: 194.206.160.4

Name : my_iseries.domain-name.com
Address: 194.206.160.112
```

Donc ici le nom de domaine correct est **domain-name.com**

Etape 2 : vérifier que le 'keytab' exporté utilise le nom de domaine correct.

Pour cela, utilisez iSeries Navigator, et se positionner dans « sécurité », puis « Service d'authentification réseau ». Ensuite par click-droit sélectionner l'entrée de menu contextuel « Manage keytab ». Cliquer ensuite sur le bouton « Details ».

Cela doit afficher une ligne avec :

Principal Type: i5/OS

Principal Name: krbsvr400/my_iseries.domain-name.com@DOMAIN-NAME.COM

Où DOMAIN-NAME.COM est le « Realm » i5/OS

Si cela est incorrect, il est nécessaire de modifier la configuration et de réexporter le keytab, et/ou vérifier la configuration DNS, afin d'obtenir correspondance des noms de domaine.

b. *Le cryptage DES droit être activé sur les comptes créés depuis le keytab.*

Sinon, l'erreur suivante apparaît : "*Encryption or checksum type is not supported.*"

Pour activer ce mode de cryptage, il faut ouvrir l'application « utilisateurs et ordinateurs Active Directory » sur le serveur sur lequel le keytab a été exporté. Sélectionner l'entrée « Users », et choisir l'utilisateur nommé comme :

my_iseries_1_krbsvr400

(Il peut y en avoir d'autres sous la même forme, comme: my_iseries_2_krbsvr400, …)
Dans les propriétés de ce utilisateur, choisir « Compte », et cocher « Utiliser les types de cryptage DES pour ce compte »

c. *Erreur de connexion : "Not authorized to access key table".*

Le fichier keytab doit être lisible depuis l'utilisateur IBM i qui est utilisé pour EASYCOMD, par défaut QTCP.

Il est nécessaire de connaître l'emplacement du fichier keytab. Pour cela, utiliser iSeries Navigator, et sélectionner « Sécurité », puis « Service d'authentification réseau ». Sur le menu contextuel, choisir « Manage keytab ». Suivre l'assistant jusqu'à la dernière étape (annuler ensuite). Le chemin du fichier keytab est afficher sur ce dernier écran d'assistant.

Cet emplacement est généralement:

```
/QIBM/UserData/OS400/NetworkAuthentication/keytab/krb5.keytab
```

Pour accorder les droits de lectures à QTCP, utiliser la commande suivante :

```
CHGAUT OBJ('/QIBM/UserData/OS400/NetworkAuthentication/keytab/krb5.keytab')
USER(QTCP) DTAAUT(*R)
```

d. ***L'horloge de toutes les machines impliquées doit être synchronisée***

Si vous obtenez des erreurs telles que '*ticket not yet valid*' ou bien '*ticket is expired*', cela est probablement dû à une non-correspondance des horloges.

Vérifiez les valeurs systèmes QTIMZON et QTIME à l'aide de la commande WRKSYSVAL, ainsi que les horloges du contrôleur de domaine, et des postes clients.

Voir aussi

L'installation de EIM sur System i

EIM avec Easycom

SSL

Connexion SSL - préalable

La connexion SSL peut utiliser un cryptage SSL.

Les conditions pour pouvoir utiliser une connexion SSL sont les suivants :

- le programme EASYCOM doit être en version 4.60.10 ou ultérieur
- le programme EASYCOMD doit être en version 3.0.3 ou ultérieur
- l'OS/400 doit être en version V5R3 ou ultérieur
- un ID d'application nommé 'EASYCOM' doit être créé sur l'OS/400 à l'aide de DCM. Un certificat doit être associé à cette application.
- Le serveur Easycom doit être configuré pour accepter les connexions SSL, via la commande CFGEAC
- Le client doit supporter SSL, disposer de l'autorité de certification ayant créé le certificat associé à l'application 'EASYCOM'.

Le support de SSL dépend des versions clientes et des plateformes utilisées. Il faut consulter la documentation des produits clients pour connaître le niveau de compatibilité SSL.

Connexion SSL - configuration du serveur

Pour activer SSL, il est nécessaire de créer une **application** et d'y associer un certificat. L'ID d'application doit être égal à 'EASYCOM'. Le certificat doit avoir été émis par une autorité de certification qui sera reconnue par le client.

Pour créer l'application, il faut utiliser **DCM (Digital Certificate Manager)**, de l'AS/400.

Remarque : Il s'agit d'une démarche identique à celle qui est nécessaire pour activer SSL pour le serveur telnet par exemple.

La configuration serveur suit les étapes suivantes :

- Se connecter à DCM en utilisant un navigateur web avec l'URL **http://my_iseries:2001,** et ensuite cliquer sur « **Gestionnaire de Certificats Numériques** ».

Si l'URL ne fonctionne pas, elle devra être activée depuis iSeries Nagivator.

- Cliquer sur « **Select a Certificate Store** », sélectionner « ***SYSTEM** », et cliquer sur « **continue** ». Il faut ensuite saisir le mot de passe du magasin de certificats et valider.
- Choisir « **manage applications** » sur le menu gauche et cliquer sur **« add application** ». Ensuite, choisir « **Serveur** » et cliquer sur « **continue** ».

Saisir « **EASYCOM** » pour l'ID d'application. Cela représente la clé qui sera utilisée par Easycom. Saisir une description et valider.

- A présent il faut **associer un certificat à l'application**. Ce certificat a pour rôle d'assurer l'identité du serveur, et également pour l'encryptage. Deux options sont possibles pour l'assignation du certificat :

1. Générer le certificat en utilisant l'autorité de certification de l'AS/400 (CA, Certificate of Authority). Dans ce cas de figure le certificat de l'autorité de certification devra être installé sur le poste client (exporter le certificat de l'autorité de certification en utilisant le menu d'exportation de DCM).
2. Demander un certificat à une autorité « tiers » déjà reconnue du poste client. Dans ce cas il faudra importer le certificat serveur dans le magasin *SYSTEM en utilisant le menu « import »

Pour associer le certificat, cliquer sur « **Manage Application** », puis sur « **Update certificate assignment** ». Choisir « **Serveur** », et cliquer sur « **continue** ». Apparaît ensuite l'association actuelle (« *none assigned* ») pour l'application.

Sélectionner l'entrée **'Easycom'** créée précédemment, et cliquer sur « **Update Certificate Assignment** ». Sélectionner le certificat approprié, et cliquer sur « **Assign New Certificate** ».

Cliquer ensuite sur « **Validate** » : cela vérifie que le certificat est valide pour ce système.

- L'étape finale consiste à configurer le serveur Easycom pour utiliser SSL à l'aide de CFGEAC :

```
CHGCURLIB EASYCOM
CFGEAC LIB(EASYCOM) SSL(*ON)
```

- Ensuite redémarrer EASYCOMD par la commande suivante:

```
STREACD PORT(*JOBD) RESTART(*YES)
```

- Ensuite tester une connexion Easycom en mode SSL. Vous pouvez utiliser l'outil Easycom Configuration à cette fin.
- La vérification des options peut s'effectuer par la commande suivante:

```
DSPMSG EASYCOM/EACMSGQ
```

Cela doit afficher:

```
   EASYCOMD:Starting from library EASYCOM, Version 3.00.03, (Nov 10 2008
11:15:49/OS530).
EASYCOMD:EASYCOM - (c)AURA Equipments - http://www.easycom-aura.com
------------------------------------ Lib=
   ;Pwd=SSL support
EASYCOMD:Configuration used for Library EASYCOM is Dq=
     SSL=On
```

En cas de problème, les erreurs apparaîtront à ce niveau.

Remarque: cela n'assure pas que la connexion a bien été effectuée en SSL, mais uniquement que la connexion SSL est supportée, et configurée.

Easycom Configuration affiche cependant cette information.

L'AS/400 peut vérifier si la connexion est effectuée en SSL ou non à l'aide du programme d'exit EACLOG002.

Pour vérifier l'état sur un travail actif, examiner la pile d'appels du travail. Pour cela, utiliser WRKACTJOB, puis l'option 5 et ensuite l'option 11. Si vous voyez « SSL_Read » dans la pile, cela signifie que la connexion est en SSL.

Connexion SSL - authentification par certificat

Easycom peut utiliser les certificats clients. Cette fonction peut être utilisée aux fins suivantes :

- plus de sécurité au niveau réseau. Le serveur peut donner l'accès uniquement aux clients disposant d'un certificat valide.
- Le certificat peut servir à déterminer l'utilisateur OS/400 à utiliser par la suite. Cela peut provenir du sujet du certificat lui-même, ou bien la base de données EIM peut définir celui-ci (le certificat est alors référencé au niveau EIM).

Le certificat client doit être valide pour l'AS/400. Un certificat est considéré valide s'il a été émis par une autorité de certification reconnue par l'AS/400 (CA), dans le magasin de certificats *SYSTEM.

Le certificat peut avoir été émis par l'AS/400. Dans ce cas l'autorité de certification est « Local CA » de *SYSTEM.

Créer un registre X.509 dans EIM, et configurer l'emplacement LDAP (optionnel).

Cette étape est nécessaire si on désire utiliser EIM pour mapper le certificat vers un utilisateur AS/400.

Dans ce cas l'utilisateur fourni par le client est « *SSL ».

Par System i Access, se positionner sur "Réseau"/"Enterprise Identity Mapping"/"Domain Management"/"<votre domaine>"/"User Registries", et cliquer sur "Add a new system registry".

Choisir un nom et "X.509" comme type de registre.

Dans "configuration", choisir « propriétés » et choisir le registre X.509 nouvellement créé.

A présent il est nécessaire de configurer l'emplacement LDAP pour le magasin de certificats *SYSTEM. Cela permettra de lier la création de certificats au registre X.509 que l'on vient de créer dans EIM.

Pour cela, utiliser le gestionnaire de certificats numériques. Se connecter à http://my_iseries:2001. Select "Digital Certificate Manager" (en V6R1 sélectionner "i5/OS management" puis "Internet configuration" first. Se connecter en QSECOFR).

Choisir "Manage LDAP location", et saisir:

LDAP server: nom de l'hôte complet : my_series.mydomain.com

Directory distinguished name (DN): dc=

Use Secure Sockets Layer (SSL): No

Port Number: 389

Login distinguished name (DN): cn=

Password: xxxx (mot de passe LDAP utilisé pour EIM).

Créer un certificat utilisateur

Aller sur https://my_iseries:2010/QIBM/ICSS/Cert/Admin/qycucm1.ndm/main0 en utilisant l'utilisateur pour lequel on veut créer le certificat client.

Choisir ensuite "Create Certificate". Le nom de login utilisé précédemment sera utilisé comme nom principal (subject) du certificat client.

Ensuite cliquer sur "install certificate". Cela installera le certificat dans le navigateur. Reste ensuite l'exporter pour pouvoir l'installer à son emplacement final (magasin de certificats Windows de l'utilisateur par exemple).

Si un registre X.509 a été créé dans EIM et si l'emplacement LDAP a été défini dans la configuration DCM, les données EIM seront automatiquement mises à jour. Remarque : l'entrée EIM **doit exister** avant cette opération (entrée avec une destination égale à cet utilisateur AS/400).

Installer le certificat client sur le magasin local

Utiliser le navigateur web pour transférer le certificat and le magasin désiré (typiquement le magasin de certificats Windows, ou encore l'emplacement OpenSSL pour Linux).

Activer la partie Easycom server pour utiliser les certificats client

CHGCURLIB EASYCOM

CFGEACAUTH LIB(EASYCOM) SSL(*ON) SSLAUTH(*ON) SSLROLE(*EIM)

Utiliser "SSLROLE(*EIM)" si un registre X.509 est utiliser ou bien *SUBJECT si le « subject » du certificat est égal au nom de l'utilisateur AS/400 à utiliser pour la connexion.

EIM doit être également configuré, à l'aide de la commande CFGEACEIM.

Essayer la connexion avec la valeur « *SSL » à la place du nom d'utilisateur, et un mot de passe vide, si EIM est utilisé. Dans le cas contraire, utiliser un nom d'utilisateur et mot de passe classique.

Si on saisit DSPMSG EASYCOM/EACMSGQ sur un terminal on doit voir apparaître un texte comme suit:

```
EASYCOMD:Starting from library EASYCOM, Version 3.00.05, (Jun 23 2009
16:29:38/OS530).
EASYCOMD:Eim connection OK - X.509 registry is 'p520 certicates'
EASYCOMD:EASYCOM - (c)AURA Equipments -
http://www.easycom-aura.com
=
EASYCOMD-V.3.00.05(EASYCOM/EASYCOMD); Lib=
Eim=
EASYCOMD:Configuration used for Library EASYCOM is Dq=
KerbAuth=
```

On y voir que le registre X.509 est détecté et nommé 'p520 certificates'.

Cela confirme la capacité SSL d'EASYCOMD.

Cela montre (à la première connexion) que la bibliothèque EASYCOM a SSL actif, et que l'authentification SSL est activer avec le rôle « *EIM ».

Les éventuels problèmes d'authentification apparaissent ici.

Les Jobs Easycom

Les jobs Easycom

Quand l'application est lancée, ou plus précisément quand une connexion s'établit avec l'AS/400, un job est créé sur l'AS/400 mais il est possible d'utiliser des prestarts jobs.

Il y a donc un job actif par instance d'application cliente connectée ou plusieurs si l'application utilise plusieurs connexions.

Création et propriétés des jobs

Le job peut être créé par le programme de sécurité EACTCP003 (voir ci-dessous),

Si EACTCP003 n'existe pas ou ne démarre pas le job, ce dernier est crée d'après:

- EACJOBD, s'il existe,
- La JOBD du profil autrement.
- La JOBD associée au profil en ce qui concerne la LIBL (voir LIBL par défaut).

EASYCOM sur l'AS/400 fonctionne par le biais d'un sous-système et d'un démon. C'est ce dernier qui va recevoir les demandes de connexion de la part des applications clientes. Quand l'application est lancée, et qu'une connexion s'établit avec l'AS/400, un job est créé sur l'AS/400. Il y a donc un job actif par instance d'application cliente connectée. Chaque application peut alors avoir ses propres ouvertures de fichier, verrous, positions courantes, transactions en cours.

EASYCOM nécessite la présence d'un sous-système EASYCOM et d'un démon EASYCOMD.

Ce démon va créer un job à la connexion et EASYCOM communique avec ce job par les APIs standards TCP/IP.

Ces jobs sont visibles par la commande WRKACTJOB, ou WRKSBSJOB sur l'AS/400.

Priorité du JOB EASYCOM

Tous les jobs sont stockés dans le sous-système EASYCOM.

Le sous-système utilise **EACJOBD** pour sa description et **EACCLS** pour sa classe de priorité.

Vous pouvez modifier la classe de priorité du sous-système avec la commande CHGCLS.

Création et propriétés des jobs

Le job peut être créé par le programme de sécurité EACTCP003 (voir ci-dessous),

Si EACTCP003 n'existe pas ou ne démarre pas le job, ce dernier est crée d'après:

- EACJOBD, s'il existe,
- La JOBD du profil autrement.
- La JOBD associée au profil en ce qui concerne la LIBL (voir LIBL par défaut).

EASYCOM sur l'AS/400 fonctionne par le biais d'un sous-système et d'un démon. C'est ce dernier qui va recevoir les demandes de connexion de la part des applications clientes. Quand l'application est lancée, et qu'une connexion s'établit avec l'AS/400, un job est créé sur l'AS/400. Il y a donc un job actif par instance d'application cliente connectée. Chaque application peut alors avoir ses propres ouvertures de fichier, verrous, positions courantes, transactions en cours.

EASYCOM nécessite la présence d'un sous-système EASYCOM et d'un démon EASYCOMD.

Ce démon va créer un job à la connexion et EASYCOM communique avec ce job par les APIs standards TCP/IP.

Ces jobs sont visibles par la commande WRKACTJOB, ou WRKSBSJOB sur l'AS/400.

Priorité du JOB EASYCOM

Tous les jobs sont stockés dans le sous-système EASYCOM.

Le sous-système utilise **EACJOBD** pour sa description et **EACCLS** pour sa classe de priorité.

Vous pouvez modifier la classe de priorité du sous-système avec la commande CHGCLS.

Niveau d'autorité du JOB "EASYCOMD"

Le programme EASYCOMD comporte des droits spéciaux. Ces droits sont nécessaire pour pouvoir assurer les fonctions de sécurité et soumettre les travaux des connexions entrantes (ou le cas échéant la connexion aux travaux à démarrage anticipé).

Le programme EASYCOMD doit avoir pour propriétaire QSECOFR et avoir l'attribut USRPRF à *OWNER, ainsi que l'utilisation des droits adoptés. EASYCOMD est lancé par défaut sous l'utilisateur QTCP mais bénéficie des droits QSECOFR grâce à ces propriétés.

Ces propriétés peuvent être restaurées en cas d'altération à l'aide des commandes suivantes :

CHGPGM PGM(EASYCOM/EASYCOMD) USRPRF(*OWNER) USEADPAUT(*YES)

CHGOBJOWN OBJ(EASYCOM/EASYCOMD) OBJTYPE(*PGM) NEWOWN(QSECOFR)

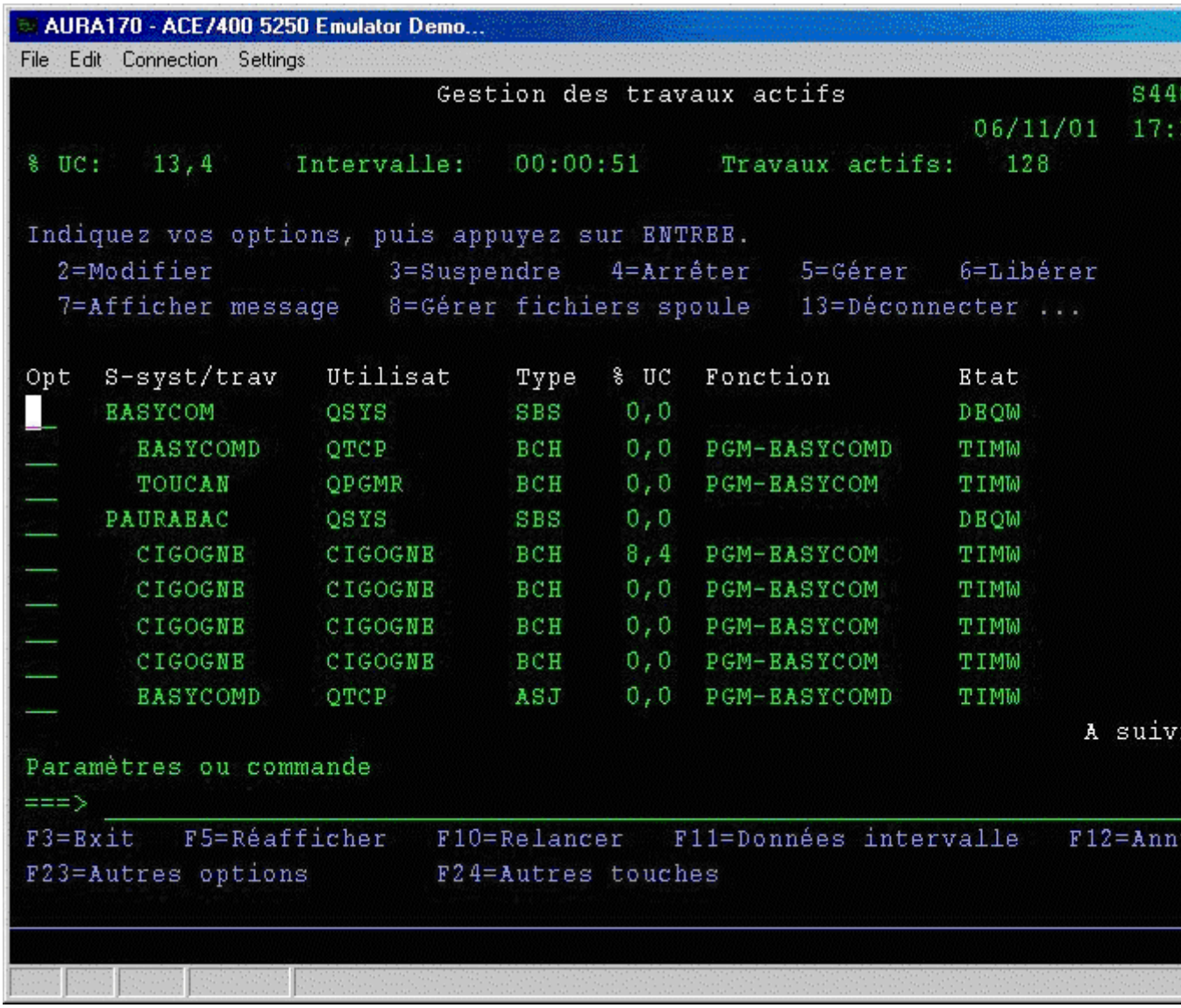
GRTOBJAUT OBJ(EASYCOM/EASYCOMD) OBJTYPE(*PGM) USER(QTCP) AUT(*USE)

Niveau d'autorité du JOB "EASYCOMD"

Par défaut, "EASYCOMD" est un job de type **QSECOFR**.

Il doit avoir l'option *ALLOBJ pour fonctionner dans des conditions optimales.

Exemple :



Dans cet exemple deux sous-systèmes tournent en parallèle (EASYCOM et PAURABAC), on retrouve le démon EASYCOMD et des jobs utilisateur dans chacun.

Les jobs EASYCOM sont autonomes, ils ont leur environnement d'exécution créé à partir de EASYCOM/ EACJOBD dans le cas d'un lien TCP/IP, ou de la JOBD de l'utilisateur pour l'APPC.

La LIBL du job est celle de l'utilisateur (voir LIBL par défaut).

Les programmes peuvent alors bénéficier des propriétés et objets propres au job tels que OVRxxx, QTEMP, *LDA, …

Remarque :

"EASYCOM" est le nom par défaut de la bibliothèque d'installation du serveur EASYCOM.

Ce nom de bibliothèque a pu être changé pendant l'installation.

Après chaque IPL de votre AS/400, le sous-système EASYCOM/EASYCOM doit être démarré :

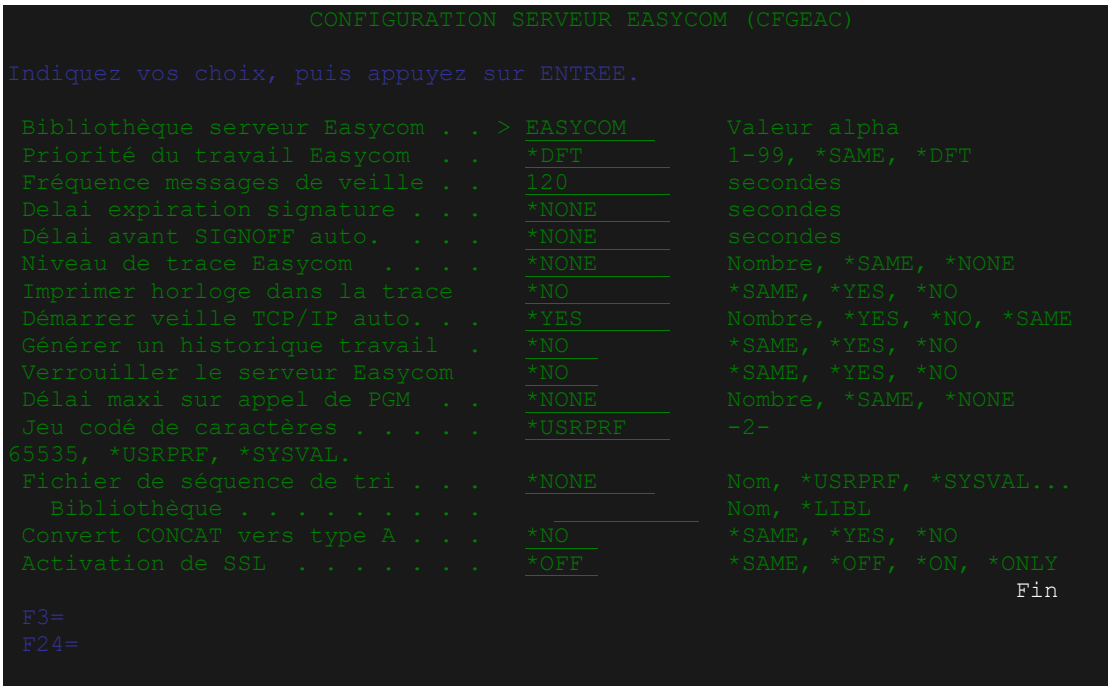
```
STRSBS EASYCOM/EASYCOM
```

Cette commande peut être incluse dans l'autostart (QSTRUP) de votre système pour le prochain IPL.

Commandes de configuration EASYCOM

CFGEAC

La commande **CFGEAC** permet de configurer des propriétés du serveur EASYCOM sur le système iSeries - AS/400.



Bibliothèque serveur EASYCOM (LIB)

Entrez le nom de la bibliothèque dans laquelle le serveur EASYCOM est installé.

Priorité du travail EASYCOM (PTY)

Ce paramètre n'est plus utilisé.

Fréquence des messages de veille (TCPTOUT)

Entrez la fréquence (en secondes) à laquelle vous voulez que le poste client envoie des messages de veille au serveur, en cas d'inactivité de l'application. La valeur spéciale 0 indique que les messages de veille ne seront pas envoyés.

En cas de coupure réseau, le travail sur l'AS/400 risque de rester actif indéfiniment. Quand l'application cliente est connectée, si elle n'échange pas des données avec le serveur, des messages de veille sont envoyés par le client vers le serveur pour indiquer à ce dernier que le réseau n'est pas coupé. Si le travail serveur ne reçoit aucun message dans un délai supérieur au délai précisé ici, le travail serveur s'arrête et annule toute transaction en cours.

Délai avant resignature (RESIGN)

Ce paramètre n'est plus utilisé.

Délai avant déconnexion (CONNECTION)

Ce paramètre n'est plus utilisé.

Niveau de trace (LOGLEV)

Une trace des requêtes échangées entre le client et le serveur peut être systématiquement faite.

La trace est générée dans le fichier LOGFILE de la librairie précisée au paramètre LIB, le membre porte le nom du travail EASYCOM associé à chaque connexion. La valeur spéciale 0 indique que la trace ne doit pas être générée.

Enregistrer l'horloge dans la trace (LOGCLOCK)

Si le niveau de trace est supérieur à 0, au paramètre LOGLEV indiquez ici si l'horodatage doit être inséré dans la trace.

Les valeurs possibles sont :

*YES : L'horodatage est inséré. Cette option est utile pour analyser les temps d'exécution des requêtes.

*NO: L'horodatage n'est pas inséré dans la trace.

Démarrer messages de veille (HBEAT)

Indique si les messages de veille doivent être attendus systématiquement, ou si le serveur doit attendre l'arrivé d'un message avant d'activer son système. Les valeurs possibles sont :

*YES: Le travail serveur s'attend à recevoir des messages de veille dés la première inactivité.

*NO: Le travail serveur attend l'arrivé du premier message de veille de la part du poste client, avant d'activer son système d'arrêt automatique en cas de coupure. Certain postes clients (Linux) ne sont pas en mesure d'envoyer des messages de veille.

Historique détaillé (JOBLOG)

Indique si un historique détaillé du travail EASYCOM doit être fait par le système. Les valeurs possibles sont :

*YES: Un historique détaillé est généré par le système.

*NO: L'historique détaillé n'est pas généré.

Verrouiller le serveur (LOCKED)

Le serveur EASYCOM peut être configuré pour ne permettre l'accès aux données qu'à des applications autorisées. Quand le serveur EASYCOM est ainsi verrouillé, le programme client doit passer un mot de passe au programme EACP003 qui autorisera l'accès aux données. Les valeurs possibles sont :

*YES : Le travail EASYCOM attend l'appel du programme EACP003 avant de permettre l'ouverture des fichiers ou l'appel de tout autre programme.

*NO (par défaut) : Le serveur EASYCOM n'a pas besoin du mot de passe applicatif pour accéder aux données.

Délai d'appel de programmes(PGMTOUT)

Quand les applications clientes appellent des programmes externes sur l'AS/400 (voir ASLanceRPC) on peut déterminer la durée maximale d'exécution de ces programmes, au delà de laquelle le serveur EASYCOM considérera que l'appel a échoué. Cette option évite d'avoir des travaux qui restent indéfiniment dans un état MSGW quand le programme appelé sort en erreur non récupérée.

ID codé de jeu de caractères (CCSID)

Indiquez ici le jeu de caractère sous lequel le travail EASYCOM doit tourner. Par défaut, le jeu de caractère est *HEX (65535).

Séquence de tri (SRTSEQ)

Indique le nom du fichier de tri à utiliser pour les comparaisons et les tris. Les valeurs possibles sont celles utilisables pour le paramètre SRTSEQ dans la commande système CHGJOB.

Convertir champs CONCAT vers type A (CONCATF)

Indiquez ici si les champs résultant d'opérations CONCAT doivent toujours être considérés comme des champs de type alphanumériques. Les valeurs possibles sont :

*YES: Les champs résultats de CONCAT seront gérés comme un seul champ de type alphanumérique.

*NO: Les champs résultats de CONCAT gardent leur type d'origine.

Activation de SSL (SSL)

Configure comment SSL doit être utilisé avec Easycom.

Utilisez DSPMSG EACMSGQ pour savoir si l'initialisation de SSL a fonctionné.

Remarque: La modification de cette option nécessite le redémarrage d'Easycomd pour être prise en compte. Vous pouvez effectuer cette opération par les commandes STRSBS/ENDSBS ou bien par la commande STREACD.

Les valeurs possibles sont:

*YES: Les connexions SSL et non-SSL sont admises.

*NO: Les connexions SSL ne sont pas utilisées. Une tentative de connexion en SSL échouera.

*ONLY: L'utilisation de SSL est obligatoire. Toute connexion non SSL échouera.

CFGEACTCP

OBSOLETE : utilisez CFGEACSBS.

Des objets sont créés au moment de l'installation, et peuvent être recréés s'il le faut par la commande **CFGEACTCP**.

Elle crée les objets requis : SBSD, JOBD, JOBQ, CLS.

EASYCOM :	*SBSD	Sous-système dans lequel les travaux clients seront démarrés.
EACJOBD :	*JOBD	Description de travail pour les travaux clients.
EACJOBQ :	*JOBQ	File d'attente de travaux pour les clients.
EACCLS :	*CLS	Classe pour les travaux clients.
EASYCOMD :	*JOBD	Description de travail pour le travail EASYCOMD, qui doit toujours être actif dans le sous-système.

La commande **CFGEACTCP** permet aussi de changer le nom du sous-système, et de changer le numéro du **port par défaut (6077).**

Attention : On ne peut créer qu'un seul sous-système par bibliothèque EASYCOM. Pour avoir plusieurs sous-systèmes, il faut dupliquer la bibliothèque, et lancer la commande CFGEACTCP dans chaque bibliothèque.

Si le port par défaut est changé, il faudra le préciser lors de la connexion par les clients (soit dans le prompt, soit dans la configuration), dans l'adresse de l'AS/400 séparé par un caractère ' : ' (Deux points).

Par exemple 194.206.160.111:6079

La commande **CFGEACTCP** permet de configurer le serveur EASYCOM pour les connexions TCP/IP.

Librairie EASYCOM (LIB)

Entrer le nom de la librairie où les objets du serveur EASYCOM se trouvent, et où le nouveau sous-système doit être créé.

Librairie système en LIBL (SYSLIB)

Ce paramètre n'est plus utilisé.

Nom du sous système EASYCOM (SBS)

Entrer le nom du sous-système à créer dans la librairie. Quand le sous-système sera actif, les travaux associés à chaque connexion tourneront dans ce sous-système.

Port du service EASYCOM (PORT)

Entrez le numéro du port TCP que vous voulez attribuer au serveur EASYCOM. Si vous voulez exécuter plusieurs serveurs EASYCOM sur la même machine, vous devez attribuer un numéro de port différent à chacun d'entre eux.

Les valeurs possibles sont :

*DFT : Si un service nommé EASYCOM existe dans la table des services, le port qui lui est associé sera utilisé. Voir la commande système WRKSRVTBLE pour gérer la table des services. Si le service EASYCOM n'existe pas, le **port par défaut 6077** est utilisé.

Nombre : Un numéro de port à allouer au nouveau serveur EASYCOM.

Autoriser travaux prédémarrés (PJ)

Utiliser les travaux pré démarrés dans le sous système. Les valeurs possibles sont :

*OFF : Les travaux pré démarrés ne sont pas utilisés quand une session cliente demande une connexion, même s'ils sont configurés sur le sous-système.

*ON / *AUTO : Utiliser les travaux pré démarrés du sous système si ceux-ci sont configurés et actifs.

CFGEACSBS (Configurer le sous-système Easycom)

Cette commande configure le sous-système EASYCOM et la façon dont les travaux client sont lancés dans le sous-système.

Nom du sous-système EASYCOM (SBS)

Entrez le nom du sous-système Easycom à configurer et son nom de bibliothèque.

Mode de démarrage par défaut des JOB (MODE)

Indique comment les JOB client Easycom sont lancés dans le sous-système lorsque le client se connecte.

Les valeurs possibles sont :

***SAME** : ne change pas le mode actuel.

***BCH** : lors de la connexion du client, un nouveau travail par lots est soumis.

***BCI** : lors de la connexion du client, un nouveau travail immédiat par lots est lancé.

Le temps de connexion est plus court en utilisant le mode *BCI.

***PJ** : Lors de la connexion Client, Easycom essaie de se connecter à une tâche pré-démarrée (pre-start job).

Si aucune tâche de pré-démarrage n'est disponible, une BCI est démarrée.

Numéro de port du service Easycom (PORT)

Le numéro de port TCP/IP que vous souhaitez allouer au serveur Easycom.

Si vous souhaitez utiliser plusieurs serveurs Easycom sur la même machine IBM i, vous devez allouer un numéro de port différent à chacun.

Les valeurs possibles sont :

***SAME** : Ne change pas la configuration du port.

***DFT** : Si un service easycom existe dans la table des services, le numéro de port associé à ce service sera utilisé.

Voir la commande système WRKSRVTBLE pour gérer la table des services.

Si le service Easycom n'existe pas, le numéro de port par défaut 6077 est utilisé.

Numéro : Un numéro de port à allouer au serveur Easycom.

Description de poste de serveur (JOBD)

Description utilisée lorsque les travaux Easycom sont soumis, lorsque MODE est défini sur *BCH.

Les valeurs possibles sont :

***SAME** : Ne modifie pas le paramètre JOBD.

***DFT** : Utiliser la description de travail EACJOBD par défaut.

***USRPRF** : Utiliser la description de poste du profil utilisateur.

Nom JobD : Définissez le nom et la bibliothèque de la description du travail à utiliser lorsque les travaux Easycom sont soumis.

Numéro des travaux de pré-démarrage initial (INLJOBS)

Spécifie le nombre initial de travaux de pré-démarrage (pre-start job) lancés au démarrage du sous-système.

Les valeurs possibles sont :

***SAME** : Ne modifie pas le réglage actuel.

***NONE** : ne démarre pas les travaux de pré-démarrage lorsque le sous-système est démarré.

1-9999 : spécifiez le nombre de travaux de pré-démarrage qui sont lancés au démarrage du sous-système.

Seuil de prédémarrage des travaux (THRESHOLD)

Spécifie quand des travaux de pré-démarrage (pre-start job) supplémentaires sont lancés.

Lorsque le pool de travaux disponibles est réduit en dessous de ce nombre, d'autres travaux sont démarrés et ajoutés au pool disponible.

Les valeurs possibles sont :

***SAME** : Ne modifie pas le réglage actuel.

1-9999 : spécifiez le nombre minimum de travaux de pré-démarrage qui doivent être disponibles avant que des travaux de pré-démarrage supplémentaires ne soient démarrés.

Travaux de pré-démarrage supplémentaires (ADLJOBS)

Spécifie le nombre supplémentaire de travaux de pré-démarrage (pre-start job) qui sont démarrés lorsque le nombre de travaux de pré-démarrage tombe en dessous de la valeur spécifiée avec le paramètre Threshold.

Les valeurs possibles sont :

***SAME** : Ne modifiez pas le réglage actuel.

1-9999 : spécifiez le nombre de travaux de pré-démarrage supplémentaires à démarrer.

Profil utilisateur de pré-démarrage des travaux (PJUSER)

Spécifie le nom du profil utilisateur sous lequel le travail de pré-démarrage (pre-start job) est lancé

Les valeurs possibles sont :

***SAME** : Ne modifie pas le réglage actuel.

QUSER : Le profil utilisateur QUSER fourni par IBM est utilisé.

name : spécifiez le nom du profil utilisateur utilisé pour le travail de pré-démarrage.

Nom des travaux de pré-démarrage (PJJOB)

Spécifie le nom du travail de pré-démarrage (pre-start job) démarré.

Les valeurs possibles sont :

***SAME** : Ne modifie pas le réglage actuel.

name : spécifiez le nom du travail de pré-démarrage.

Prédémarrage du travail JOBD (PJJOBD)

Spécifie le nom qualifié de la description de travail utilisée pour le travail de prédémarrage (pre-start job).

Les valeurs possibles sont :

***SAME** : Ne modifie pas le réglage actuel.

***DFT** : la description de travail par défaut EACJOBD est utilisée pour démarrer les travaux de pré-démarrage.

name : spécifiez le nom de la description de travail utilisée pour ce travail de pré-démarrage et son nom de bibliothèque.

Démarrer les travaux de pré-démarrage avec Sbs (PJSTART)

Spécifie si les travaux de pré-démarrage (pre-start job) doivent être lancés au démarrage du sous-système.

Les valeurs possibles sont :

***SAME** : Ne modifie pas le réglage actuel.

***OUI** : Les travaux de pré-démarrage sont lancés au démarrage du sous-système.

***NO** : Les travaux de pré-démarrage ne sont pas lancés au démarrage du sous-système.

STREACD

La commande **STREACD** démarre le service EASYCOM. Le programme EASYCOMD est démarré dans le sous-système pour permettre la connexion des stations clientes.

Librairie EASYCOM (LIB)

Entrer le nom de la librairie où les objets du serveur EASYCOM se trouvent, et où la description du sous-système a été créée.

Port du service EASYCOM (PORT)

Entrez le numéro du port TCP que vous voulez attribuer au serveur EASYCOM. Si vous voulez exécuter plusieurs serveurs EASYCOM sur la même machine, vous devez attribuer un numéro de port différent à chacun d'entre eux. Les valeurs possibles sont :

*DFT : Si un service nommé EASYCOM existe dans la table des services, le port qui lui est associé sera utilisé. Voir la commande système WRKSRVTBLE pour gérer la table des services. Si le service EASYCOM n'existe pas, le port par défaut 6077 est utilisé.

*JOBD : Le service est démarré selon la description de travail EASYCOMD dans la librairie.

Nombre : Un numéro de port à allouer au nouveau serveur EASYCOM.

Autoriser travaux pré démarrés (Prestarts jobs - PJ)

Ce paramètre n'est utilisé que si le paramètre PORT est différent de *JOBD. Utiliser les travaux pré démarrés dans le sous système. Les valeurs possibles sont :

*OFF (par défaut) : Les travaux pré démarrés ne sont pas utilisés quand une session cliente demande une connexion, même s'ils sont configurés sur le sous-système.

*ON / *AUTO : Utiliser les travaux pré démarrés du sous système si ceux-ci sont configurés

et actifs.

Redémarrer travail EASYCOMD (RESTART)

Redémarrer le travail EASYCOMD si celui ci est déjà actif dans le sous système. Les valeurs possibles sont :

*NO : Si le programme EASYCOMD est déjà actif, il reste inchangé.

*YES : Si le programme EASYCOMD est déjà actif, il est arrêté, puis démarré avec les nouveaux paramètres.

Le sous-système EASYCOM doit être démarré **après le démarrage de TCP/IP**.
Le travail EASYCOMD (démon) tourne en permanence dans le sous-système EASYCOM. Il est démarré automatiquement au démarrage du sous-système.

Pour accepter les demandes de connexion des postes clients, EASYCOM utilise le **port 6077**.

Si un système de sécurité ou une autre application interdit l'utilisation de ce numéro de port il peut être modifié par la commande CFGEACSBS.

EACINSTALL (Easycom Install)

C'est une commande utilisée en fin d'installation. Cette commande met à jour les objets Easycom pour des fonctionnalités optimales en fonction de la version de system i installée.

Actuellement cela influence sur l'interface SQL par défaut à utiliser ainsi que le support d'EIM pour EASYCOMD.

```
                              EASYCOM INSTALLATION (EACINSTALL)

Type choices, press Enter.

Easycom Library . . . . . . . . EASYCOM Lib. of product EASYCOM
OS VERSION FOR ADJ.  . . . . . .   *AUTO          MINIMUM OS VERSION FOR ADJ
LEVEL OF SQL INTERFACE TO USE  .   *AUTO          *CISC, *EMBED, *CLI, *AUTO
```

Il est possible de sélectionner LEVEL of SQL interface to use de *CLI à *EMBED. Cela permet à Easycom d'utiliser l'interface 'SQL encapsulé' au lieu de l'interface CLI. L'interface CLI est plus riche en fonctionnalités mais utiliser *EMBED peut aider à résoudre des problèmes rencontrés avec l'interface CLI. L'interface *CISC est obsolète et n'est plus inclue dans les versions récentes d'Easycom.

Les limitations de l'interface *EMBED sont : non support des LOB ou procédures stockées SQL. Cependant dans certains cas cela est plus performance que CLI.

En fait, *EMBED est l'ancienne – historique – interface est *CLI est l'actuelle. Seulement l'interface *CLI bénéficiera d'améliorations futures.

CFGEACAUTH (Authentification)

Cette commande permet de choisir quelles méthodes d'authentification sont utilisées.

```
                     Authentification Easycom (CFGEACAUTH)

Indiquez vos choix, puis appuyez sur ENTREE.

Bibliothèque serveur Easycom . . > EASYCOM Valeur alphanum
Utiliser SSL . . . . . .   *ON           *SAME, *OFF, *ON, *ONLY
Utiliser authentification SSL  .   *ON           *SAME, *OFF, *ON, *ONLY
```

```
Role authentification SSL  . . .   *EIM
Utiliser auth. Kerberos . . . .   *OFF          *SAME, *OFF, *ON, *ONLY
```

Utiliser SSL

Cette option définit si SSL est supporté, ou obligatoire. Les valeurs possibles sont les suivantes :

*OFF: SSL n'est pas utilisé par le serveur Easycom.

*ON: SSL est utilisé s'il est demandé par le client

*ONLY: SSL est obligatoire. La connexion échoue si le client ne supporte pas SSL ou si la négociation SSL échoue.

Utiliser authentification SSL

Cette option définit si l'authentification SSL par certificat client est supportée.

*OFF: Authentification SSL par certificat non supporté. Le certificat client, s'il est fourni, ne sera pas utilisé.

*ON: Authentification SSL par certificat est supportée. La validité du certificat (s'il est fourni) est vérifiée. La connexion est abandonnée si le certificat n'est pas valide.

*ONLY: Un certificat client, valide, doit obligatoirement être fourni.

Rôle authentification SSL

Cette option définit comment sera utilisé le certificat, en plus de servir à authentifier la connexion.

*NONE: le certificat client ne sert qu'à sécuriser la connexion. Un utilisateur et mot de passe AS/400 doit être fourni en plus ; ou alors une authentification Kerberos/EIM.

*EIM: Easycom utilise EIM pour trouver une correspondance depuis le certificat client vers un utilisateur AS/400. L'activation de la configuration EIM est dans ce cas nécessaire.

*SUBJECT: le champ « subject » du certificat est égal au nom de l'utilisateur AS/400. Seul le certificat est utilisé pour toute la sécurité de la connexion.

Utiliser auth. Kerberos

Cette option détermine (indépendamment de SSL) si l'authentification Kerberos est acceptée. Pour mener à un utilisateur AS/400, EIM doit être également configuré.

CFGEACEIM (Enterprise Identity Mapping)

Cette commande permet de configurer la connexion EIM pour Easycom. Elle remplace la commande CFGEASSO qui est à présent obsolète.

Le système EIM sert à déterminer le nom d'un utilisateur AS/400 à partir d'un autre type d'authentification.

EIM peut trouver l'utilistateur AS/400 à partir des sources suivantes :

- à partir d'une authentification Kerberos. Cela permet le single signon (SSO).
- à partir d'un certificat client lors d'une connexion SSL avec certificat.

La commande CFGEACAUTH définit quels types d'authentification sont valides (Kerberos, SSL ou les deux).

```
                    Easycom EIM Configuration (CFGEACEIM)

 Indiquez vos choix, puis appuyez sur ENTREE.

Bibliothèque serveur Easycom . . > EASYCOM       Valeur alphanum
 Utiliser EIM dans Easycom  . . . > *YES          *YES, *NO, *SAME
 EIM valide de  . . . . . .   *NONE          HHMM =
 EIM valide à . . . . . .   *NONE          HHMM =
 EIM: utilisateur LDAP  . . . . .   'administrator'

 EIM: mot de passe LDAP . . . . .


 Login EIM obligatoire  . . . . .   *NO           *YES, *NO
 dn LDAP pour EIM . . . . . .
service spn pour EIM . . . . . .
```

Utiliser EIM dans Easycom

Activer ou non EIM pour Easycom. Si *NO, aucune autre option n'est visible.

EIM valide de à

EIM n'est actif que dans ces plages horaires. Pour plus de réglages concernant les périodes d'utilisation utiliser l'exit program EACLOG002.

Utilisateur LDAP pour EIM

Utilisateur à utiliser pour la connexion locale à LDAP. Cette information est nécessaire pour pouvoir déterminer l'utilisateur i5/OS correspondent à la connexion par Kerberos.

Ce nom d'utilisateur est le nom spécifier dans le logiciel IBM iSeries Navigator lors de la configuration EIM (en sélectionnant NetWork/EIM Domain Mapping/Domain Management/<yourDomain>).

Il faut spéficier le nom d'utilisateur uniquement, sans "cn=

LDAP password for EIM

Mot de passé correspondent à l'utilisateur LDAP.

Login EIM onligatoire

Configure Easycom pour interdire toute connexion non EIM (avec utilisateur/mot de passe i5/OS)

Dn LDAP pour EIM

Syntaxe alternative pour spécifier les informations de connexion à LDAP. Le paramètre Utilisateur LDAP doit être vide si cette option est utilisée. Exemple de valeur possible :

cn=

service spn pour EIM

Permet de spécifier un nom de service principal spécifique. Si *DFT est précisé, Easycom utiliser "krbsvr400" et le nom du système.

Exemple de valeurs valides (systemi5 est le nom du système, testdomain.com est le nom de domaine et TESTDOMAIN.COM est le nom de 'realm') :

krbsvr400/systemi5

krbsvr400/systemi5@TESTDOMAIN.COM

krbsvr400/systemi5.testdomain.com@TESTDOMAIN.COM (valeur par défaut si *DFT est précisé)

Exit Programs

Exit Programs

EASYCOM propose quelques programmes appelés « Exit Programs ».

Ces programmes, suivant une spécification précise, et alimentés par l'administrateur système, permettent un contrôle plus avancé et protégé des connexions easycom.

Certains devront être obligatoirement écrits comme ceux liés au Single Sign On ou au verrouillage d'Easycom (Lock EASYCOM Host).

D'autres sont liés à une configuration particulière mais restent facultatifs comme pour le cas de l'utilisation des Prestarts Jobs.

Les autres sont facultatifs et destinés à renforcer la sécurité.

Des sources d'exemple sont présents dans le fichier source **EACSYSSRC** de la bibliothèque Easycom.

Démarrage d'Easycom

Démarrage du travail client - EACSTART

Si un programme nommé EACSTART se trouve dans la liste de bibliothèques (LIBLE) du job, ce programme est appelé à chaque soumission de job par le programme EASYCOM .

Il est notamment utile pour définir des propriétés ou faire des opérations de maintenance.

A ce moment l'utilisateur est connu et le job est initialisé, ce programme peut modifier des attributs ou paramètres mais ne peut pas annuler le job.

Utiliser plutôt EACTCP003 pour contrôler les droits de l'utilisateur et éventuellement annuler le job.

Prestart job init - EACPJINI

Si les Prestarts Jobs sont activés dans la configuration EASYCOM serveur et si un programme nommé EACPJINI se trouve dans la liste de bibliothèques du job , ce programme est appelé à chaque démarrage de Prestart Job.

Il permet notamment de définir des propriétés pour le job, ou bien de préparer l'environnement, mais à ce moment l'utilisateur connecté n'est pas connu. L'avantage est de pouvoir réduire le temps de connexion.

SQL init - EACSQLINI

Ce programme est appelé lorsqu'SQL est initialisé pour la première fois dans le travail Easycom, avant toute utilisation de SQL (entre l'initialisation de SQL et la première utilisation réelle de SQL par exemple la préparation d'une requête SQL).

Lors de l'utilisation de prestart jobs**, le programme est appelé avant l'établissement de la connexion** (SQL est systématiquement initialisé avant la connexion afin de réduire le temps de connexion). Dans le cas contraire le programme est appelé lors de la première utilisation de SQL (il peut donc je jamais être appelé si SQL n'est jamais utilisé).

Cet exit program peut servir à contrôler l'environnement à ce stade.

Login et contrôle d'accès

Contrôle de connexion - EACTCPP01

Cet exit program est destiné à contrôler la connexion avant même la phase d'authentification.

Il peut également contrôler si la connexion doit s'établir en SSL ou non.

```
PGM PARM(&LIB &TPNAME &RMTADDR &IPVERSION +
&SSLASK &SSLCNF &VALID)
     DCL VAR(&LIB) TYPE(*CHAR) LEN(10)
     DCL VAR(&TPNAME) TYPE(*CHAR) LEN(30)
     DCL VAR(&RMTADDR) TYPE(*CHAR) LEN(50)
     DCL VAR(&IPVERSION) TYPE(*CHAR) LEN(1)
     DCL VAR(&SSLASK) TYPE(*CHAR) LEN(1)
     DCL VAR(&SSLCNF) TYPE(*CHAR) LEN(1)
     DCL VAR(&VALID) type(*CHAR) len(10)
```

&LIB est la bibliothèque le programme Easycom réside

&TPNAME est le nom du programme Easycom. Est égal à Easycom par défaut.

&RMTADDR est l'adresse TCP/IP du client. Elle peut être en syntaxe IPv4 ou IPv6 selon la valeur de &IPVERSION

&IPVERSION est égal à 4 ou 6 selon la version de TCP/IP

&SSLASK informe si le client essaie d'initier une connexion de type SSL. Les valeurs possibles sont :

- 'Y' : le client supporte SSL, et si le serveur l'accepte, la connexion utilisera SSL. La connexion sera donc **peut-être** en SSL.
- 'N' : le client ne supporte pas SSL ou ne demande pas à l'utiliser. La connexion ne sera pas en SSL en aucun cas.

&SSLCNF informe si le serveur supporte ou non SSL. Cette valeur est modifiable. Les valeurs possibles sont :

- 0 : le serveur n'utilisera pas SSL, même si supporté
- 1 : le serveur pourra utiliser SSL si SSLASK=Y. Si la négociation SSL échoue, la connexion demeurera valide
- 3 : le serveur doit utiliser SSL. Si SSLASK=N ou si la négociation SSL échoue, la connexion sera abandonnée.

&VALID est utilisé pour informer EASYCOMD d'autoriser ou interdire la connexion en cours. Les valeurs possibles sont :

- *YES : la connexion peut se poursuivre
- *DENY : la connexion en cours est abandonnée immédiatement. Un message d'erreur sera affiché sur le poste client.

Remarque : le programme peut modifier les paramètres &SSLCNF et &VALID.

Contrôle de login - EACLOG002

EACLOG002 est un exit program pour le processus général d'authentification.

Ce programme est appelé **après** l'authentification effectuée par Easycom.

Il est appelé quel que soit le type d'authentification en cours (normal, SSO, ou EIM).

Il peut être utilisé pour auditer l'utilisation d'Easycom et/ou refuser des connexions suivant des critères personnalisés.

EACLOG001 est la version précédente de EACLOG002 ; il ne sera pas appelé si EACLOG002 est présent.

EACLOG002 a deux paramètres supplémentaires par rapport à EACLOG001, concernant la version IP et le mode SSL.

Le prototype est le suivant:

```
PGM PARM(&LOGTYPE &RC &LOGUSER &LOGDOMAIN &USER
&IPADDR &STATION)

DCL VAR(&LOGTYPE) TYPE(*CHAR) LEN(10)
DCL VAR(&RC) TYPE(*CHAR) LEN(10)
DCL VAR(&LOGUSER) TYPE(*CHAR) LEN(130)
DCL VAR(&LOGDOMAIN) TYPE(*CHAR) LEN(130)
DCL VAR(&USER) TYPE(*CHAR) LEN(10)
DCL VAR(&IPADDR) TYPE(*CHAR) LEN(130)
DCL VAR(&STATION) TYPE(*CHAR) LEN(130)
DCL VAR(&IPVERSION) TYPE(*CHAR) LEN(1)
DCL VAR(&SSL) TYPE(*CHAR) LEN(1)
```

&LOGTYPE est en entrée et indique le type du logon en cours. Les valeurs possibles sont les suivantes:

*STD: il s'agit d'un logon standard (utilisateur et mot de passé i5/OS). &LOGUSER et &LOGDOMAIN ne sont pas renseignés.

*EIM: il s'agit d'un logon de type EIM. Le mot de passe n'est pas renseigné. &LOGUSER, &LOGDOMAIN et &USER sont renseignés.

*SSO: il s'agit d'un login de type SSO Easycom. Tous les autres paramètres sont renseignés.

&RC est le résultat de la commande. Cela peut indiquer que la connexion doit être refusée ou bien que le nom de l'utilisateur i5/OS a été modifié par le programme.

Les valeurs possibles sont les suivantes :

*OK: le logon est accepté

*CHG: le paramètre &USER a été modifié par le programme. Remarque : &USER ne subira pas de validation de mot de passe (la validation initiale fait foi).

*OUTOURS: le logon sera rejeté pour cause de non respect de la plage horaire.

*DENY: le logon sera rejeté.

&LOGUSER est le nom d'utilisateur Windows. Ce paramètre n'est renseigné que si &LOGTYPE est égal à *EIM ou *SSO.

&LOGDOMAIN est le nom de domaine Windows. Ce paramètre n'est renseigné que si &LOGTYPE est égal à *EIM ou *SSO.

&USER est le nom d'utilisateur i5/OS. Le travail Easycom tournera sous les droits de cet utilisateur.

&IPADDR est l'adresse IP du client. Ce paramètre peut servir à l'audit ou à un filtrage.

&STATION est la chaîne de caractères correspondant au nom de la station du client. Cela peut représenter un nom réel de machine (le nom correspondant à une adresse IP) ou bien le nom d'un terminale si la connexion est déclenchée par l'intermédiaire d'une connexion de type RDP.

&IPVERSION est égal à 4 ou 6 selon la version tcp/ip utilisée pour la connexion (IPv4 ou IPv6)

&SSL est égal à 'Y' si la connexion utilise SSL et 'N' sinon. La négociation SSL est déjà effectuée à ce stade.

Sécurité par restriction - EACTCP003

Il est également possible de restreindre l'utilisation d'EASYCOM à un groupe d'utilisateurs et/ou à un groupe de PC. Pour ce faire, il faut écrire un programme de validation nommé EACTCP003 dans la bibliothèque d'EASYCOM.

Si un programme EACTCP003 se trouve dans la liste de bibliothèques d'EASYCOMD, il sera appelé à chaque tentative de connexion d'une application cliente, sauf si EASYCOM est configuré pour utiliser les Prestarts Jobs (dans ce cas voir EACTCP002).

Ce programme peut laisser le client soumettre le job selon les règles habituelles en affectant *YES à la variable &JOBNAME, il peut refuser la connexion en retournant *NO dans la variable &JOBNAME ou bien soumettre lui même le job et ses propriétés et retourner son nom dans la variable &JOBNAME.

Squelette du programme :

```
PGM PARM(&TPPGM &TPLIB &USER &EAC_PARM1 + &EAC_PARM2 &RMT_ADR
&JOBNAME)
DCL  VAR(&TPPGM) TYPE(*CHAR) LEN(10)
DCL  VAR(&TPLIB) TYPE(*CHAR) LEN(10)
DCL  VAR(&USER) TYPE(*CHAR) LEN(10)
DCL  VAR(&EAC_PARM1) TYPE(*CHAR) LEN(30)
DCL  VAR(&EAC_PARM2) TYPE(*CHAR) LEN(30)
DCL  VAR(&RMT_ADR) TYPE(*CHAR) LEN(50)
DCL  VAR(&JOBNAME) TYPE(*CHAR) LEN(10)
```

Paramètres :

TPPGM : Target Program Name

TPLIB : Bibliothèque contenant le programme TPPGM

Nom et bibliothèque du programme serveur qui sera exécuté, "EASYCOM" par défaut.

USER : Nom de l'utilisateur

Nom de l'utilisateur de la nouvelle connexion cliente (peut être utilisé pour restreindre les accès à un groupe d'utilisateurs).

EAC_PARM1 :Paramètre 1 du programme TPPGM

Paramètre à transmettre si vous effectuez vous-même le SBMJOB.

EAC_PARM2 :Paramètre 2 du programme TPPGM

Paramètre à transmettre si vous effectuez vous-même le SBMJOB.

RMT_ADR : Adresse TCP/IP de l'appelant

Adresse TCP/IP de l'appelant (peut désigner un ensemble de stations de travail).

JOBNAME : Nom du job pour SBMJOB

Nom du job situé dans le sous-système EASYCOM , par défaut c'est le nom du poste client.

Si le JOBNAME a la valeur *NO, le job EASYCOM ne sera pas démarré.

Si le JOBNAME a la valeur *YES, le job EASYCOM sera pris en compte, mais il devra être démarré par le programme EACTCP003.

Exemple

EACTCP003 peut vérifier les droits de l'utilisateur et de son adresse TCP/IP, et effectuer les opérations suivantes :

Accorder l'exécution :

EACTCP003, laisser la variable &JOBNAME inchangée , ou remplacer par un autre nom de job, ensuite quitter normalement.

Refuser l'exécution :

Dans le programme EACTCP003, modifier la valeur du &JOBNAME à '*NO', et quitter normalement.

```
CHGVAR VAR(&JOBNAME)  VALUE('*NO')
```

Exécuter le programme cible

Dans EACTCP003, lancer le programme cible à l'aide d'un SBMJOB et modifier la valeur de la variable &JOBNAME à '*YES'.

```
CHGVAR VAR(&JOBNAME) VALUE('*YES')
SBMJOB CMD (CALL PGM(&TPLIB/&TPPGM)+
PARM (&EAC_PARM1&EAC_PARM2))
JOBD(&TPLIB/EACJOBD) +
USER(&USER) RTGDTA(*JOBD)
```

Contrôle prestart job - EACTCP002

EACTCP002 fonctionne de la même manière que EACTCP003 lorsque les Prestarts Jobs sont activés.

Puisque le job est déjà initialisé, EACTCP002 ne crée pas directement le job mais permet d'autoriser ou de refuser son démarrage ou de faire des contrôles ou des traitements avant l'initialisation.

Vérrouillage de Easycom - EACP003

En plus de la sécurité de base, on peut utiliser la sécurité au **niveau programme**.

Seuls les programmes validés par le service informatique pourront être utilisés sur l'AS/400.

Des programmes EASYCOM non habilités pourront se connecter à l'AS/400 par EASYCOM, mais ne pourront faire aucune opération (ouverture de fichier, appel de programme ou autre).

Un programme habilité devra envoyer à EASYCOM un mot de passe spécial. Un programme AS/400 réalisé par le service informatique renvoie une information pour savoir si le mot de passe est accepté. Ce mot de passe peut par exemple correspondre à la codification du programme EASYCOM.

Activer le contrôle :

Si l'entrée **'Lock Easycom host'** est positionnée à *YES dans le CFGEAC, aucun fichier ne peut être ouvert, aucun programme ne peut être appelé, aucune commande ne peut être envoyée à l'AS/400 par EASYCOM, jusqu'à ce qu'une application PC le débloque en lui envoyant un mot de passe.

Sous WinDev, il s'agit de l'option UNLOCK des Infos étendues de la connexion. Par exemple :

<EASYCOM>UNLOCK=HELLO</EASYCOM>

Cette option nécessite l'écriture d'un script nommé EACP003. Ce script doit se trouver dans la *LIBL du job EASYCOM.

Attention, si vous activez l'option et que ce script n'existe pas, EASYCOM restera bloqué et aucun job ne pourra être créé.

Voici le squelette de ce script :

```
PGM PARM(&PASSW &RESULT)
DCL VAR(&PASSW) TYPE(*CHAR) LEN(100)
DCL VAR(&RESULT) TYPE(*CHAR) LEN(10)
...
/* IF PASSW HAS THE RIGHT VALUE */
CHGVAR VAR(&RESULT) VALUES('*YES')
...
/* IF PASSW DOES NOT HAVE THE RIGHT VALUE */
CHGVAR VAR(&RESULT) VALUES('*NO')
```

Il reçoit un unique paramètre en entrée (le mot de passe applicatif, &PASSW qui n'est pas le mot de passe du profil) et retourne le paramètre &RESULT.

Une valeur à *YES autorise le démarrage du job et la poursuite des traitements.

Une valeur à *NO bloque le job.

2. Sécurité par restriction - EACTCP003

Il est également possible de restreindre l'utilisation d'EASYCOM à un groupe d'utilisateurs et/ou à un groupe de PC. Pour ce faire, il faut écrire un programme de validation nommé EACTCP003 dans la bibliothèque d'EASYCOM.

Si un programme EACTCP003 se trouve dans la liste de bibliothèques d'EASYCOMD, il sera appelé à chaque tentative de connexion d'une application cliente, sauf si EASYCOM est configuré pour utiliser les Prestarts Jobs (dans ce cas voir EACTCP002).

Ce programme peut laisser le client soumettre le job selon les règles habituelles en affectant *YES à la variable &JOBNAME, il peut refuser la connexion en retournant *NO dans la variable &JOBNAME ou bien soumettre lui même le job et ses propriétés et retourner son nom dans la variable &JOBNAME.

Squelette du programme :

```
PGM PARM(&TPPGM &TPLIB &USER &EAC_PARM1 + &EAC_PARM2 &RMT_ADR
&JOBNAME)
DCL  VAR(&TPPGM) TYPE(*CHAR) LEN(10)
DCL  VAR(&TPLIB) TYPE(*CHAR) LEN(10)
DCL  VAR(&USER) TYPE(*CHAR) LEN(10)
DCL  VAR(&EAC_PARM1) TYPE(*CHAR) LEN(30)
DCL  VAR(&EAC_PARM2) TYPE(*CHAR) LEN(30)
DCL  VAR(&RMT_ADR) TYPE(*CHAR) LEN(50)
DCL  VAR(&JOBNAME) TYPE(*CHAR) LEN(10)
```

Paramètres :

TPPGM : Target Program Name

TPLIB : Bibliothèque contenant le programme TPPGM

Nom et bibliothèque du programme serveur qui sera exécuté, "EASYCOM" par défaut.

USER : Nom de l'utilisateur

Nom de l'utilisateur de la nouvelle connexion cliente (peut être utilisé pour restreindre les accès à un groupe d'utilisateurs).

EAC_PARM1 :Paramètre 1 du programme TPPGM

Paramètre à transmettre si vous effectuez vous-même le SBMJOB.

EAC_PARM2 :Paramètre 2 du programme TPPGM

Paramètre à transmettre si vous effectuez vous-même le SBMJOB.

RMT_ADR : Adresse TCP/IP de l'appelant

Adresse TCP/IP de l'appelant (peut désigner un ensemble de stations de travail).

JOBNAME : Nom du job pour SBMJOB

Nom du job situé dans le sous-système EASYCOM , par défaut c'est le nom du poste client.

Si le JOBNAME a la valeur *NO, le job EASYCOM ne sera pas démarré.

Si le JOBNAME a la valeur *YES, le job EASYCOM sera pris en compte, mais il devra être démarré par le programme EACTCP003.

Exemple

EACTCP003 peut vérifier les droits de l'utilisateur et de son adresse TCP/IP, et effectuer les opérations suivantes :

Accorder l'exécution :

EACTCP003, laisser la variable &JOBNAME inchangée , ou remplacer par un autre nom de job, ensuite quitter normalement.

Refuser l'exécution :

Dans le programme EACTCP003, modifier la valeur du &JOBNAME à '*NO', et quitter normalement.

```
CHGVAR VAR(&JOBNAME)  VALUE('*NO')
```

Exécuter le programme cible

Dans EACTCP003, lancer le programme cible à l'aide d'un SBMJOB et modifier la valeur de la variable &JOBNAME à '*YES'.

```
CHGVAR VAR(&JOBNAME) VALUE('*YES')
SBMJOB CMD (CALL PGM(&TPLIB/&TPPGM)+
PARM (&EAC_PARM1&EAC_PARM2))
JOBD(&TPLIB/EACJOBD) +
USER(&USER) RTGDTA(*JOBD)
```

EACTCP002

EACTCP002 fonctionne de la même manière que EACTCP003 lorsque les Prestarts Jobs sont activés.

Puisque le job est déjà initialisé, EACTCP002 ne crée pas directement le job mais permet d'autoriser ou de refuser son démarrage ou de faire des contrôles ou des traitements avant l'initialisation.

Prestart job init - EACPJINI

Si les Prestarts Jobs sont activés dans la configuration EASYCOM serveur et si un programme nommé EACPJINI se trouve dans la liste de bibliothèques du job , ce programme est appelé à chaque démarrage de Prestart Job.

Il permet notamment de définir des propriétés pour le job, ou bien de préparer l'environnement, mais à ce moment l'utilisateur connecté n'est pas connu. L'avantage est de pouvoir réduire le temps de connexion.

Démarrage du travail client - EACSTART

Si un programme nommé EACSTART se trouve dans la liste de bibliothèques (LIBLE) du job, ce programme est appelé à chaque soumission de job par le programme EASYCOM .

Il est notamment utile pour définir des propriétés ou faire des opérations de maintenance.

A ce moment l'utilisateur est connu et le job est initialisé, ce programme peut modifier des attributs ou paramètres mais ne peut pas annuler le job.

Utiliser plutôt EACTCP003 pour contrôler les droits de l'utilisateur et éventuellement annuler le job.

EACSSO001

Il s'agit de l'Exit Program associé à l'activation du Single Sign On en mode easycom. Il n'est pas appelé en mode EIM.

Lorsque le Single Sign On est configuré et activé (voir CFGEACEIM) et si le programme EACSSO001 se trouve dans la liste des bibliothèques du job, ce programme s'exécute sur différents événements :

- avant de mémoriser une signature (connexion simple ou session Windows)
- au moment de l'enregistrement (connexion simple ou session Windows)

- puis à chaque demande de connexion.

Paramètres

Le programme est appelé par EASYCOM en lui passant différents paramètres et retourne

&OP - Opération : événement à l'origine de l'appel du programme

**BEFORE et *WINBEFORE*

Avant de mémoriser la signature simple ou de session, le programme peut :

- modifier le nom d'utilisateur et/ou le mot de passe
- autoriser ou refuser la mémorisation

**SIGNON et *WINSIGNON*

Mémorisation de la signature, le programme :

- ne peut plus modifier l'utilisateur ou le mot de passe,
- il peut autoriser ou refuser la mémorisation

**REQUEST*

Demande de connexion, le programme :

- ne peut plus modifier l'utilisateur ou le mot de passe,
- peut effacer la mémorisation et forcer l'utilisateur à se signer de nouveau.

&RC - Retour

*OK : accepter la signature

*DENY : pour refuser la signature

*EXPIRED : la durée de validité de la signature est dépassée

*OUTHOURS : la demande est en dehors des heures autorisées,

*CHG : changement d'utilisateur

&USER / &USERLEN - nom et longueur de l'utilisateur

&PWD / &PWDLEN - mot de passe et longueur

&SOTIME - Heure au format HHMMSS

&SODATE - Date au format CYYMMDD

&IDADR - adresse IP du client

&STATION - station de travail (différent de &computer si on utilise TSE)

&COMPUTER - nom de l'ordinateur.

&LOGDOMAIN - domaine Windows

&LOGUSER - utilisateur Windows

Les variables en gras (sauf &OP) peuvent être modifiées par le programme :

&RC (autoriser ou refuser la signature ou la connexion, changer d'utilisateur, expiration ou hors horaires autorisés), &USER et &PWD pour un changement d'utilisateur.

Consulter le fichier EACSSO001 de la bibliothèque EASYCOM pour un exemple et des spécifications plus détaillées.

Single signon type Easycom - EACSSO001

Il s'agit de l'Exit Program associé à l'activation du Single Sign On en mode easycom. Il n'est pas appelé en mode EIM.

Lorsque le Single Sign On est configuré et activé (voir CFGEACEIM) et si le programme EACSSO001 se trouve dans la liste des bibliothèques du job, ce programme s'exécute sur différents événements :

- avant de mémoriser une signature (connexion simple ou session Windows)

- au moment de l'enregistrement (connexion simple ou session Windows)
- puis à chaque demande de connexion.

Paramètres

Le programme est appelé par EASYCOM en lui passant différents paramètres et retourne

&OP - Opération : événement à l'origine de l'appel du programme

**BEFORE et *WINBEFORE*

Avant de mémoriser la signature simple ou de session, le programme peut :

- modifier le nom d'utilisateur et/ou le mot de passe
- autoriser ou refuser la mémorisation

**SIGNON et *WINSIGNON*

Mémorisation de la signature, le programme :

- ne peut plus modifier l'utilisateur ou le mot de passe,
- il peut autoriser ou refuser la mémorisation

**REQUEST*

Demande de connexion, le programme :

- ne peut plus modifier l'utilisateur ou le mot de passe,
- peut effacer la mémorisation et forcer l'utilisateur à se signer de nouveau.

&RC - Retour

*OK : accepter la signature

*DENY : pour refuser la signature

*EXPIRED : la durée de validité de la signature est dépassée

*OUTHOURS : la demande est en dehors des heures autorisées,

*CHG : changement d'utilisateur

&USER / &USERLEN - nom et longueur de l'utilisateur

&PWD / &PWDLEN - mot de passe et longueur

&SOTIME - Heure au format HHMMSS

&SODATE - Date au format CYYMMDD

&IDADR - adresse IP du client

&STATION - station de travail (différent de &computer si on utilise TSE)

&COMPUTER - nom de l'ordinateur.

&LOGDOMAIN - domaine Windows

&LOGUSER - utilisateur Windows

Les variables en gras (sauf &OP) peuvent être modifiées par le programme :

&RC (autoriser ou refuser la signature ou la connexion, changer d'utilisateur, expiration ou hors horaires autorisés), &USER et &PWD pour un changement d'utilisateur.

Consulter le fichier EACSSO001 de la bibliothèque EASYCOM pour un exemple et des spécifications plus détaillées.

Sécurité des objets et programmes

EACSOPEN - Ouverture Fichier et SQL

Ce programme d'exit est appelé à chaque fois qu'un fichier est ouvert ou une requête SQL est préparée (ou exécutée directement).

Le programme d'exit peut refuser l'opération ou bien changer soit le mode d'ouverture soit le texte SQL.

Un source d'exemple est visible dans le fichier source EACSYSSRC de la bibliothèque Easycom.

EACRCMD - Remote command

Ce programme d'exit est appelé à chaque exécution de commande déclenchée via Easycom.

Le programme d'exit peut refuser l'exécution, ou modifier la commande.

Un source d'exemple est visible dans le fichier source EACSYSSRC de la bibliothèque Easycom.

EACSCALL - appel de programmes

Ce programme d'exit est appelé à chaque exécution d'un programme ou d'une procédure de programme de services.

Le programme d'exit peut refuser l'appel, ou bien modifier le nom du programme ou de la procédure.

Un source d'exemple est visible dans le fichier source EACSYSSRC de la bibliothèque Easycom.

EACSIFS - Accès IFS

Ce programme est appelé pour chaque accès IFS.

Les paramètres sont le chemin du fichier et le mode d'ouverture. Le mode d'ouverture est un numérique combinaison des constantes suivantes (hexadécimales):

_EAC_IFSOPEN_READ=1 accès en lecture

_EAC_IFSOPEN_WRITE=2 accès en écriture

_EAC_IFSOPEN_CREAT=4 le fichier sera créé si inexistant

_EAC_IFSOPEN_EXCL=8 le fichier doit ne pas exister avant appel (création obligatoire)

_EAC_IFSOPEN_TRUNC=10 troncature du fichier

_EAC_IFSOPEN_APPEND=20 écriture en fin de fichier

_EAC_IFSOPEN_BINARY=40 mode binaire

_EAC_IFSOPEN_BIGFILE=80 gros fichier. Permet l'ouverture de fichiers > 2 Go

En mode création:

_EAC_IFSMODE_RUSR 400 utilisateur peut lire (u+r)

_EAC_IFSMODE_WUSR 800 utilisateur peut écrire (u+w)

_EAC_IFSMODE_XUSR 1000 utilisateur peut exécuter (u+x)

_EAC_IFSMODE_RGRP 2000 groupe peut lire (g+r)

_EAC_IFSMODE_WGRP 4000 groupe peut écrire (g+w)

_EAC_IFSMODE_XGRP 8000 groupe peut exécuter (g+x)

_EAC_IFSMODE_ROTH 10000 autres peut lire (o+r)

_EAC_IFSMODE_WOTH 20000 autres peut écrire (o+w)

_EAC_IFSMODE_XOTH 40000 autres peut exécuter (o+x)

Mode de partage:

_EAC_IFSSHARE_RDONLY 100 0000 partage en lecture seule

_EAC_IFSSHARE_WRONLY 200 0000 partage en écriture

_EAC_IFSSHARE_NONE 400 0000 aucun partage (exclusif)

_EAC_IFSSHARE_RDWR 300 0000 partage en lecture/écriture

Pour pouvoir tester le mode d'ouverture, il faut effectuer un ET binaire avec la valeur à tester et vérifier que le résultat est égal à cette valeur.

Le programme d'exit peut refuser l'accès au fichier.

Un source d'exemple est visible dans le fichier source EACSYSSRC de la bibliothèque Easycom.

Erreurs d'installation

Erreur de connexion 11001 (2AF9 Hexa) : Host not found

L'AS/400 peut être désigné par un nom sur le réseau TCP/IP, au niveau du DNS ou du fichier host. Cette erreur survient lorsque ce nom est utilisé à la place de l'adresse IP dans les paramètres de la connexion et qu'il n'est pas trouvé et associé à la bonne adresse IP.

Vérifier le nom de l'AS/400, le fichier host, le(s) serveur(s) DNS, ou utiliser une adresse IP de type xxx.xxx.xxx.xxx

Où se trouve le fichier Hosts ?

En général dans le répertoire C:\WINDOWS\system32\drivers\etc.

Ce fichier contient les correspondances des adresses IP aux noms d'hôtes. # Chaque entrée doit être sur une ligne propre. L'adresse IP doit être placée dans la première colonne, suivie par le nom d'hôte correspondant. L'adresse IP et le nom d'hôte doivent être séparés par au moins un espace.

De plus, des commentaires peuvent être insérés sur des lignes propres ou après le nom d'ordinateur. Ils sont indiqué par le

symbole '#'.

Par exemple :

194.206.10.1 main.as # serveur AS400 principal

194.206.10.2 test.as # serveur AS400 de test

194.206.10.100 serveur.info1

194.206.10.101 poste_x

....

Serveur DNS

Vérifier l'adresse du serveur DNS dans les propriétés de Protocole Internet (TCP/IP) de votre connexion réseau.

Erreur de connexion 10060 (274C Hexa) : Connection Time out

L'adresse TCP/IP contactée n'existe pas sur le réseau.

Vérifier l'adresse TCP/IP de l'AS/400 ou son nom.

Si un nom de machine AS/400 est utilisé, vérifier qu'il est bien référencé sur les serveurs DNS.

Si vous êtes sûr de l'adresse IP, c'est sûrement le fait d'un firewall.

Si vous avez un firewall qui protège votre AS/400, le port 6077 doit être ouvert.

Erreur de connexion 10061 (274D Hexa) : Connection Refused

Vérifier l'adresse IP, ou le nom de l'AS/400.

Vérifier que le sous-système EASYCOM a bien été démarré sur l'AS/400.

Lancer le sous-système par la commande :

```
STRSBS EASYCOM/EASYCOM
```

Si le sous-système est bien démarré, vérifier que le travail EASYCOMD tourne.

Sinon, démarrer le travail EASYCOMD par la commande :

```
STREACD EASYCOM
```

Ou, arrêter le sous-système, et le relancez.

Tester la connexion par l'outil de configuration ou d'administration de EASYCOM.

Si le travail EASYCOMD ne démarre jamais, consulter les éventuels messages générés par EASYCOM par les commandes suivantes :

```
DSPMSG EASYCOM/EACMSGQ
```

Ou

```
DSPPFM EASYCOM/LOGFILE
```

EASYCOM utilise par défaut le port numéro 6077.

Si ce numéro est déjà utilisé, configurer EASYCOM pour en utiliser un autre.

Easycom écrit des informations dans le fichier **LOGFILE** à chaque problème de connectivité TCP/IP, notamment.

- Coupure non programmé. Par exemple si l'utilisateur coupe son processus windows par le gestionnaire de tâche, ou coupure réseau franche.

- Partie PC plantée pendant plus de 2 minutes (cas comprenant également câble débranché par exemple). C'est le 'heartbeat timeout'.

Le temps de 120 secondes est réglable par CFGEAC.

Easycom Configuration

Easycom Configuration est l'outil de gestion centralisé côté PC de l'accès natif Easycom.

La configuration de la partie serveur se fait depuis un terminal ou un émulateur.

Toutes les options sont stockées dans les fichier easycom.ini.

Ce fichier de paramètre peut être global (répertoire de Windows) ou spécifique à une application (répertoire de l'exécutable).

Cet utilitaire se présente sous la forme de 6 onglets:

- Paramètres de connexion
- Activation de clé EASYCOM
- Gestion des fichiers de trace
- Préférences Easycom
- Sécurité
- Vérification des versions de modules (DLL)

Paramètres de connexion

Nom ou adresse IP de l'AS/400

Sélectionnez cette option et renseignez le nom ou l'adresse TCP/IP de la machine.

L'utilisation du nom implique une configuration DNS ou du fichier hosts.

Serveur Easycom

- **Défaut (EASYCOM/EASYCOM)**

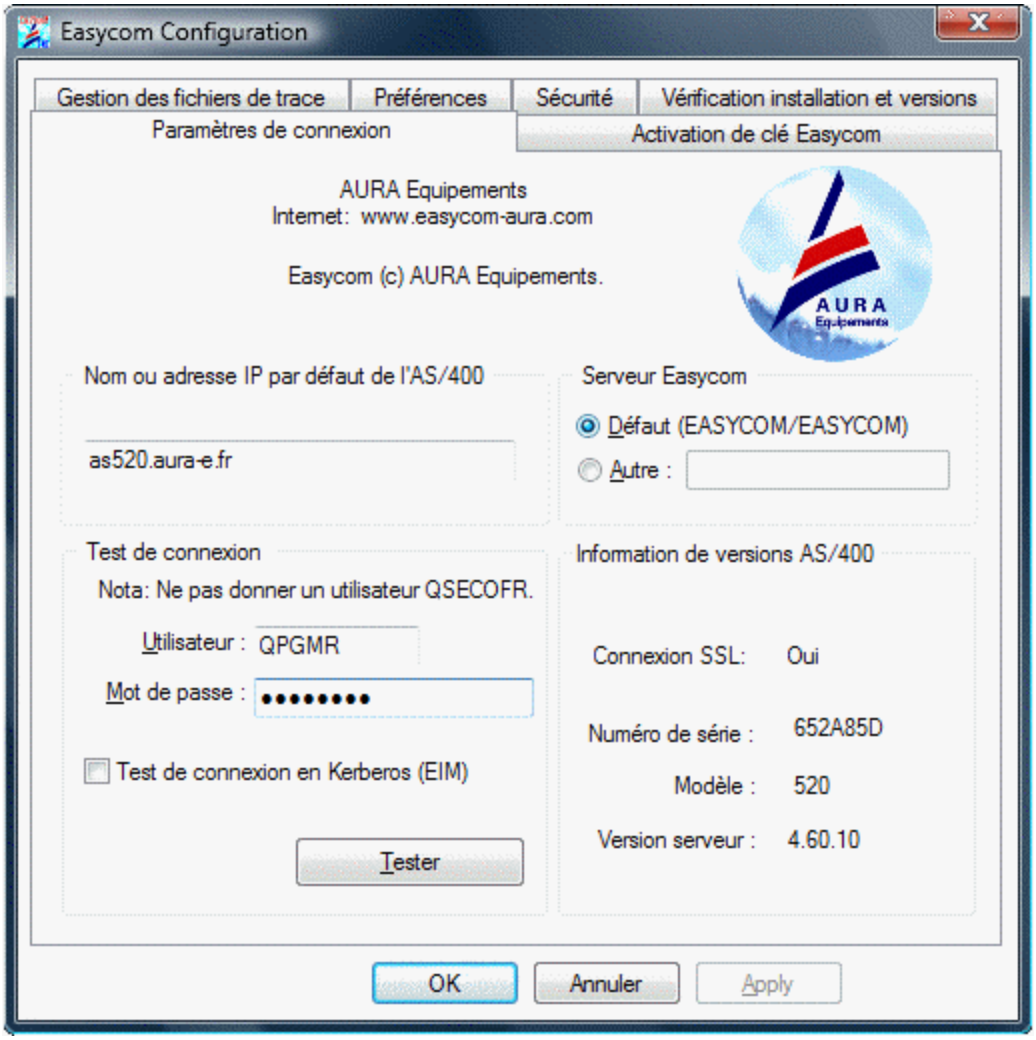
  Utilisez le programme Serveur par défaut : (EASYCOM/EASYCOM)

- **Autre**

  Sélectionnez le programme du serveur à lancer (BIBLIOTHEQUE/PROGRAMME) au cours de la connexion. Le programme du serveur est le programme AS/400 démarré par le routeur ou démarré par le job EASYCOMD.

Test de connexion

Ces options ne sont utilisées que pour le test de connexion et **ne sont pas** mémorisées.



Saisir un utilisateur/mot de passe, ou cliquer sur « Test de connexion en Kerberos (EIM) », puis cliquer sur le bouton **"Tester".**

Si la connexion est réussie, les informations de versions AS/400 s'affichent.

Numéro de série : 650643C

Modèle : 1520

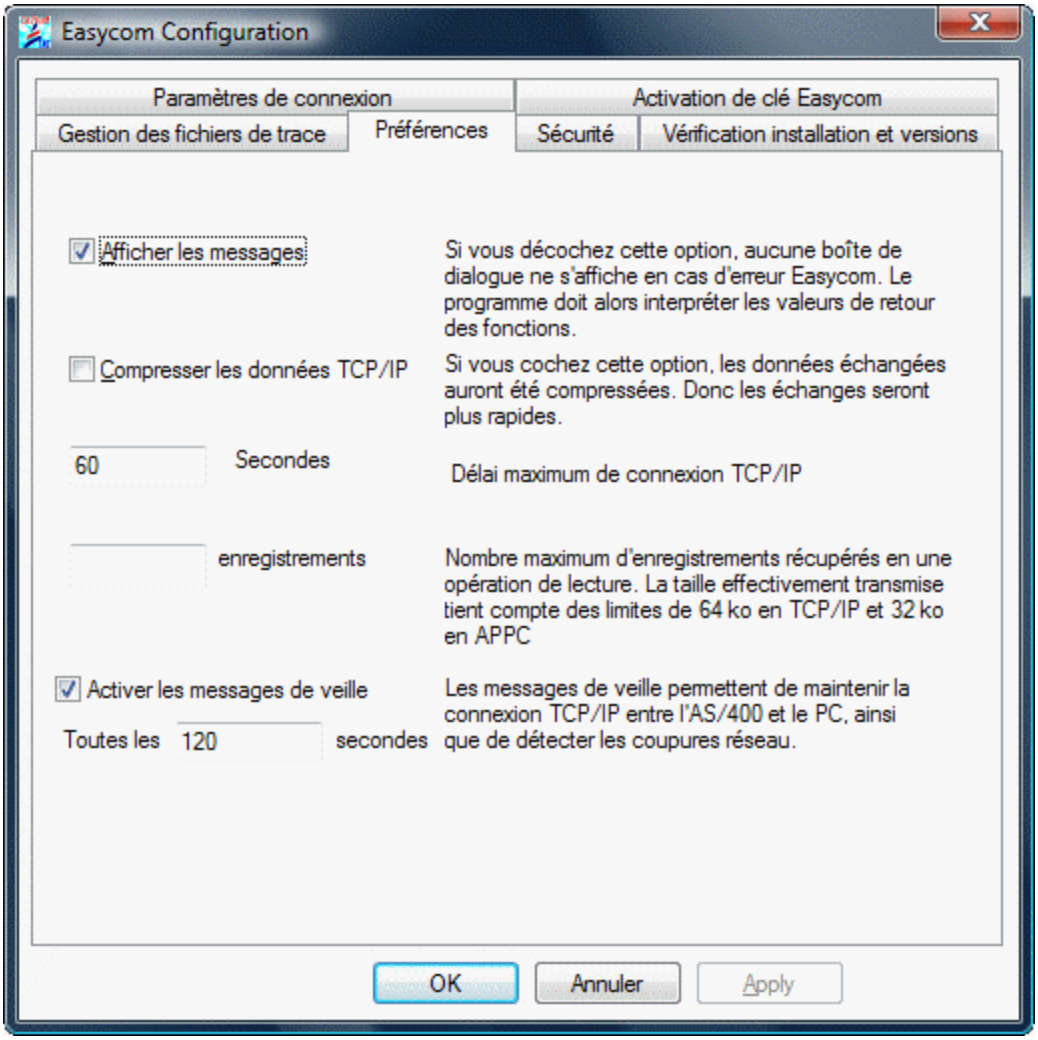
Version serveur : 4.60.10

Si SSL a été activé il est indiqué si la connexion a réellement été établie en SSL ou non

Si la connexion Kerberos a été choisie, le nom de l'utilisateur OS/400 réel est indiqué.

Préférences

Cette partie vous permet de paramétrer EASYCOM afin d'optimiser au mieux les temps d'accès réseau ou bien de modifier certaines préférences de fonctionnement.



Afficher les messages

En décochant cette option, aucune boîte de dialogue ne s'affichera en cas d'erreur EASYCOM. Le programme devra alors interpréter les valeurs de retour des fonctions dans tous les cas (par exemple : erreur de mot de passe). Cette option est conseillée pour un programme PC de type "serveur" (web ou tout programme qui fonctionne sans intervention).

On peut également désactiver les boites de dialogues par l'option SHOWDIALOGS=0 des infos étendues de la connexion.

Compresser les données TCP/IP

Cette option permet d'utiliser une compression des données, ce qui permet de réduire les volumes échangés entre l'AS/400 et le PC.

Délai maximum de connexion TCP/IP

Par défaut : 60 secondes.
Timeout = vide : Valeur par défaut de 60s

Timeout = 0 : Attente indéfinie

Nombre maxi d'enregistrements récupérés

Cette option permet de fixer le nombre maximum d'enregistrements lus en un bloc. Sa valeur par défaut est 32, et elle est limitée par le paramètre réglant la taille d'un bloc.

Il est également possible de définir un cache en lecture (et même en écriture) avec la fonction ASPropriété.

Activer les messages de veille

Dans le cas d'utilisations prolongées des applications sans qu'aucun échange de données ne soit effectué, il est possible que TCP/IP coupe la communication entre l'AS/400 et le PC. Pour éviter cela, il est possible d'envoyer à intervalles réguliers des messages sur la connexion établie.

Cela permet aussi une terminaison automatique du job EASYCOM en cas de non réponse prolongée du PC.

Clé d'activation - détail

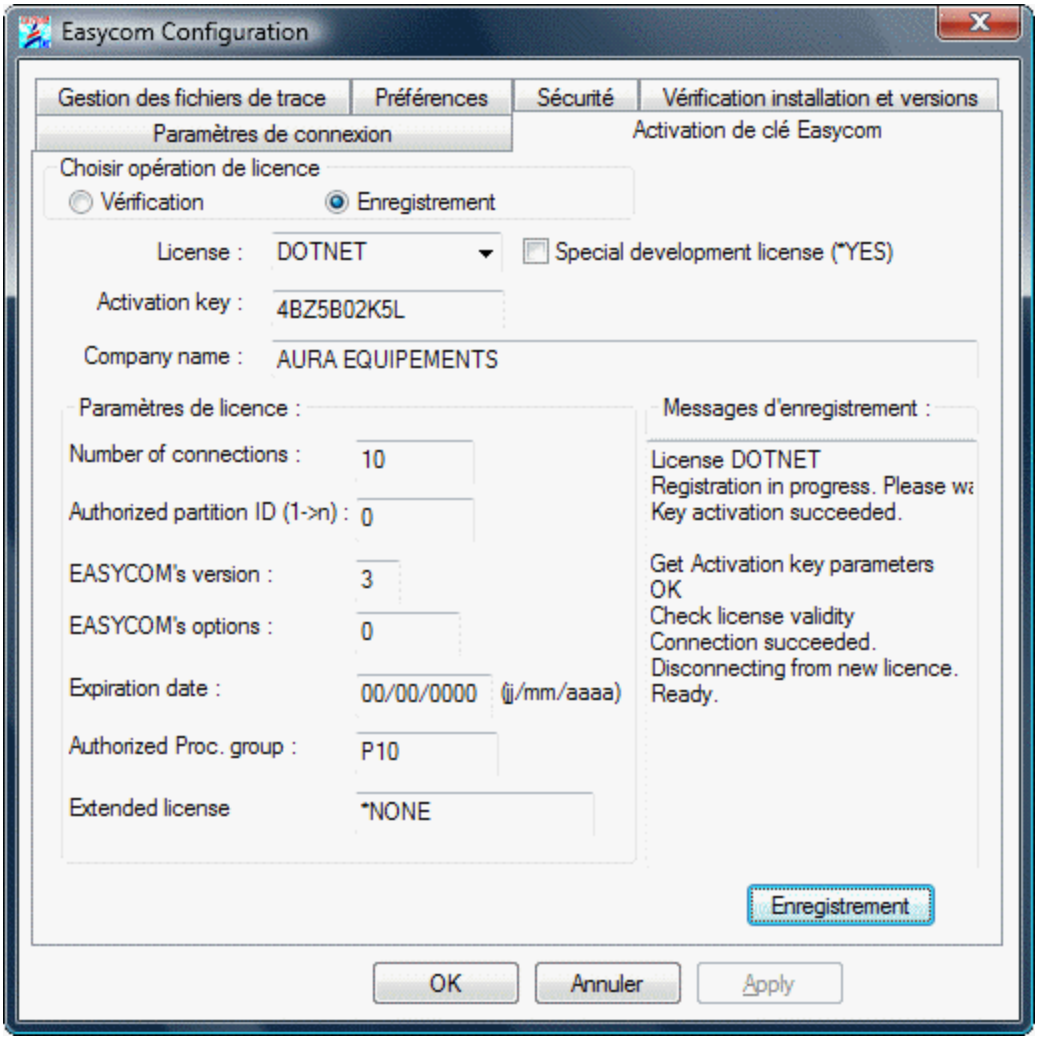
La quasi-totalité des problèmes ou erreurs lors de l'activation de la clé proviennent de fautes de frappe dans un des champs à renseigner. En cas d'erreur, vérifier attentivement chaque valeur (licence, nom de la société...).

La clé d'activation est notamment calculée à partir du numéro de série de l'AS/400, pour connaître ce numéro vous pouvez taper la commande `DSPSYSVAL SYSVAL(QSRLNBR)` sur un terminal ou utiliser le test de connexion depuis Easycom Configuration.

Depuis le PC

Depuis le PC, ouvrir Easycom Configuration, dans l'onglet "Activation de la clé Easycom".

Cet écran permet de vérifier ou de saisir une clé, cocher l'opération souhaitée et sélectionner dans la liste la version à vérifier ou à saisir. Voir le détail des champs ci-dessous.



Depuis l'AS400

Sur un terminal AS/400, faire **CHGCURLIB EASYCOM** puis **EASYREG.**

La commande EASYREG permet d'enregistrer les droits de licence d'utilisation des produits de AURA Equipements sur une machine iSeries - AS/400.

Saisir les paramètres de clé communiqués par Aura Equipements dans l'écran suivant :

```
License . . . . . .    WINDEV11
Special Development License        *YES ou *NO
Company name . . . . . .    Nom_de_votre_Société
Activation Key . . . . . .    Votre_clé
Number of connections . . . . .    selon la licence
Authorized partition ID (1->n)     0
EASYCOM'S Version . . . . . .    3
EASYCOM'S Options . . . . . .    0
Expiration date (ddmmyyyy) . . .    00000000 (ou valeur pour
```

```
                                    une clé temporaire)
Authorized Proc. group . . . .      *
Extended licence                    *NONE
Use for development only . . . .    *YES ou *NO
```

Détails des champs

License / Licence d'utilisation (LICENCE)

Entrez le nom de la licence d'utilisation tel qu'il vous est donné dans le document fourni par AURA Equipements ou votre revendeur.

Special Development License / Licence de développement (DEVPT)

Indique si la licence est accordée pour le développement d'applications, ou leur déploiement seul.

Les valeurs possibles sont :

*YES : La licence est accordée pour le développement des applications et leur maintenance ou debogage.

*NO : Seul le déploiement des applications sera autorisé.

Company name / Société licenciée (COMPANY)

Nom ou raison sociale de la société ou personne à qui la licence d'utilisation est accordée.

Recopiez exactement le texte tel qu'il est rédigé dans le document fourni par AURA Equipements ou votre revendeur.

Activation Key / Clé d'activation (KEY)

Entrez le code d'activation tel qu'il est imprimé dans le document remis par AURA Equipements ou votre revendeur.

Number of connections / Nombre de sessions simultanées (CONNECTION)

Cette valeur représente le nombre maximum de sessions qui pourront se connecter simultanément sous la licence spécifiée par le paramètre LICENCE. La valeur spéciale 0 indique un nombre illimité de sessions.

Authorized partition ID (1->n) / Numéro de partition autorisée (STATION)

Sur un système partitionné, si la licence n'est accordée qu'à une seule partition, entrez ici le numéro de cette partition, tel qu'il est précisé sur le document qui vous a été remis.

La valeur spéciale 0 indique que la licence est accordée pour n'importe quelle partition.

EASYCOM'S Version / Version de licence (VERSION)

EASYCOM'S Options / Options de la licence (OPTION)

Usage interne: Recopiez la valeur telle qu'elle est imprimée sur le document qui vous a été remis.

Expiration date (ddmmyyyy) / Date d'expiration (DATE)

Le droit de licence peut vous être accordé provisoirement, dans le cadre d'un prêt, d'une location, ou dans l'attente d'un évènement. Entrez ici la date d'expiration, sous la forme JJMMAAAA, telle qu'elle est imprimée dans le document qui vous a été remis. La valeur spéciale 00000000 indique que la licence n'est pas limitée dans le temps.

Authorized Proc. group / Groupe de processeur (MODEL)

Si la licence d'utilisation est accordée pour un niveau maximum de groupe de processeur iSerie - AS/400, entrez ici ce groupe, tel qu'il est imprimé dans le document qui vous a été remis.

La valeur spéciale * indique que la licence est accordée pour tous les groupes de processeurs.

Pour connaître le groupe de votre système, exécutez la commande WRKLICINF.

Extended licence / Extension de licence (EXTLIC)

Des options complémentaires ou limitatives peuvent être données ici. Notamment pour une licence WinDev Mobile (option WDMOB). Recopiez le texte tel qu'il est imprimé dans le document qui vous a été remis.

Pour développement seulement (DEVONLY)

Indiquez ici si cette licence sera utilisable pour un déploiement d'application, dans le cas ou le nombre maximum autorisé pour le déploiement est atteint. Cette option ne s'applique que sur les licences de développement, quand le paramètre DEVPT a la valeur *YES.

Les valeurs possibles sont :

*YES : La licence sera utilisable pour un déploiement d'application, quand le nombre maximum de sessions accordées par la licence de déploiement est atteint.

*NO : Cette licence ne sera pas utilisable pour le déploiement d'applications. Elle est réservée aux développeurs uniquement.

Attention : Différencier O (lettre) de 0 (chiffre) et I (lettre) de 1 (chiffre).

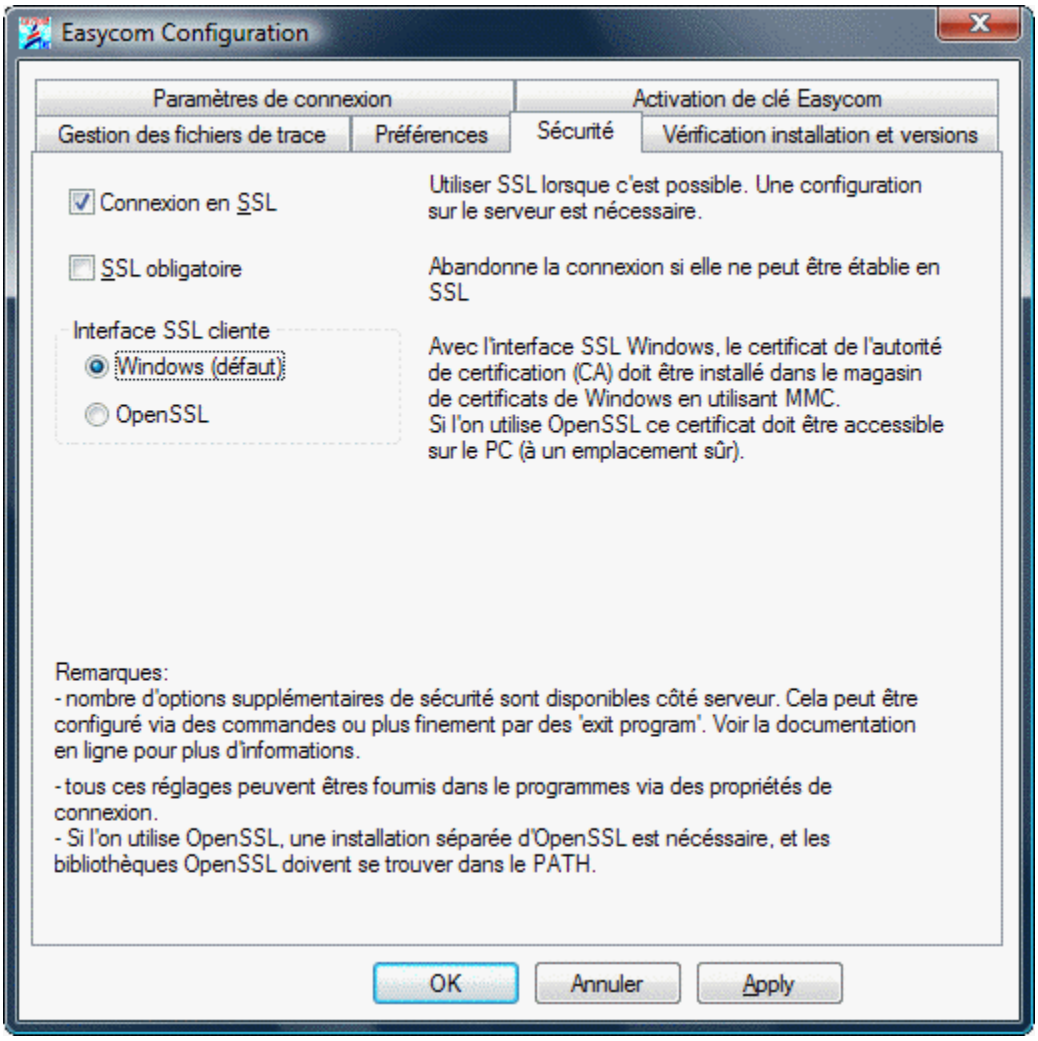
Erreur en validant la clé d'activation

Si la validation de la clé retourne l'erreur 1, les données ne sont pas correctes ; vérifier bien chaque champ.

Si la validation de clé retourne l'erreur 6, arrêter tous les jobs Easycom en cours, recommencez.

Sécurité

Cette page concerne les options SSL à appliquer par défaut à la connexion. Ces options SSL peuvent également être configurées au niveau de l'application cliente elle-même.



Si SSL est activé, la connexion sera tentée en SSL. Si la négociation SSL échoue ou n'est pas supportée par le serveur, le processus de connexion continuera sans SSL (sans cryptage).

Si SSL est activé et obligatoire, la connexion ne sera établie que si la négociation SSL a fonctionné.

Remarques :

- si la partie cliente n'est pas à jour, ces options peuvent être ignorées, et la connexion pourra fonctionner sans SSL
- si la partie cliente est à jour mais pas la partie serveur, la connexion sera abandonnée si SSL est configuré comme obligatoire.
- La page de test de connexion indique si la connexion a pu utiliser SSL avec succès ('Oui', 'Non', ou 'N/A' si pas supporté par le client). Si ce test a fonctionné, cela ne garantit pas que l'application pourra utiliser SSL, étant donné que la partie client est spécifique à chaque produit (Delphi, WinDev, PHP, …)

Cette page de configuration propose deux interfaces :

- Windows (défaut). Utilisation l'interface SSL intégrée à Microsoft Windows. Il peut être nécessaire d'installer le certificat de l'autorité de certification (CA, Certificate Authority) qui a émis le certificat du serveur SSL (voir Connexion SSL - configuration du serveur)

  Pour cela, utiliser mmc.exe (Microsoft Management Console), et ajouter le plugin de gestion de magasin de certificats. Pour cela, dans le menu démarrer, faire « exécuter », puis taper « certmgr.msc », et entrée. Ensuite, clic-droit sur « Autorités de certification racines de confiance », puis choisir « toutes les tâches », et « Importer ». Il faut ensuite sélectionner le fichier qui contient le certificat

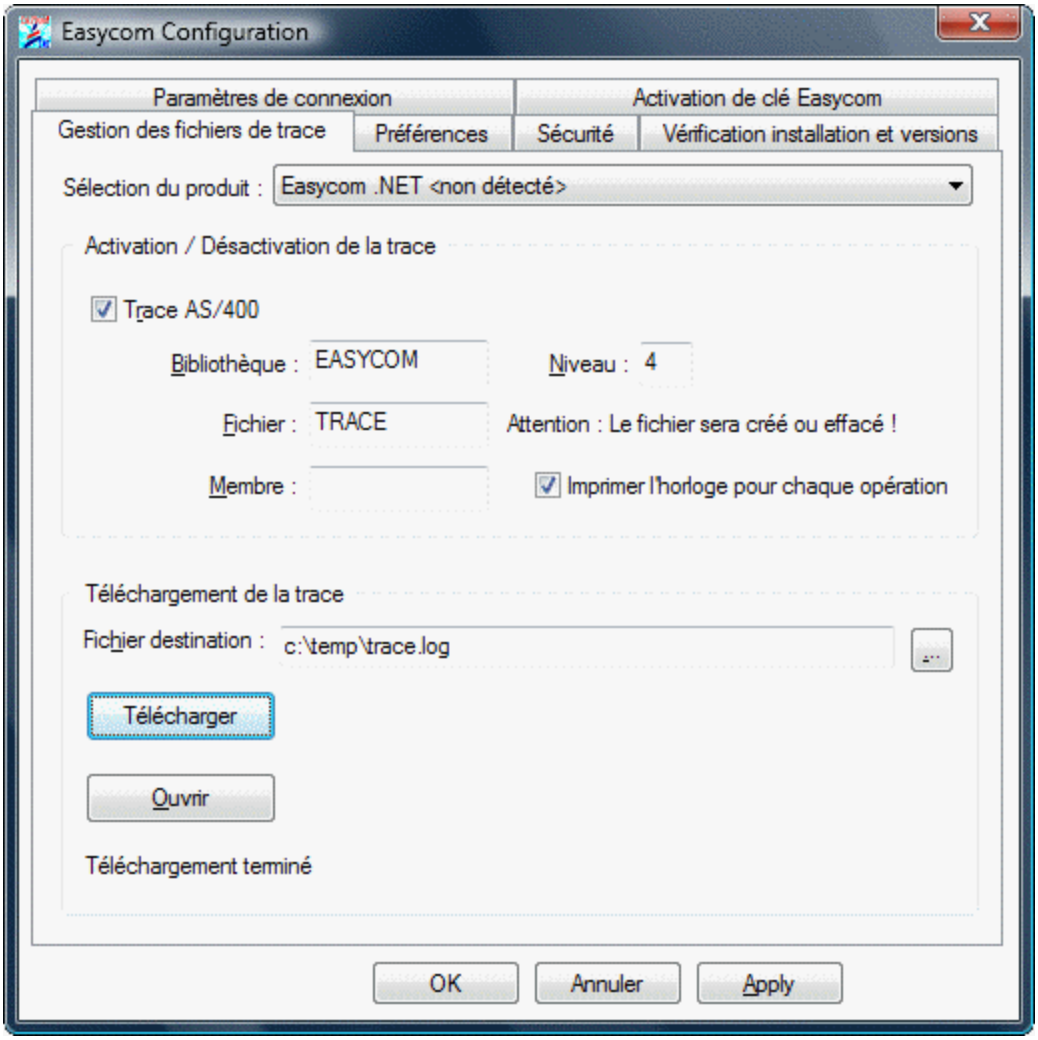
- OpenSSL. Utilisation de l'interface OpenSSL. Dans ce cas de figure les bibliothèques d'OpenSSL doivent être présentes sur le PC. Il est également nécessaire de disposer du certificat de l'autorité de certification. Il est nécessaire de spécifier le chemin ou le nom du fichier (format .pem), dans Easycom Configuration ou bien au niveau de l'application cliente.

Gestion des fichiers de trace

Parfois en cas d'erreur ou à des fins d'audit des opérations effectuées par EASYCOM, il peut être intéressant d'effectuer des traces de ce qui est fait par votre programme.

Pour cela, il existe le mode de trace AS/400.

Le mode trace réduit significativement les performances, il doit être strictement réservé à des fins d'analyse.



Activation / Désactivation de la trace

- **Trace AS/400 :**

  Sélectionnez **"Trace AS/400"** pour activer ou désactiver la trace générée par Easycom sur l'AS/400.

- **Bibliothèque :**

  Si vous voulez utiliser la trace AS/400, il vous faudra spécifier au moins un nom de bibliothèque et un nom de fichier sur l'AS/400.

  Faites attention à donner un nom de bibliothèque dans laquelle vous avez les droits d'écriture.

- **Fichier :**

  Le fichier sera créé s'il n'existe pas puis il sera écrasé par la suite.

  Vous pouvez également plus ou moins détailler les commandes envoyées à l'AS/400.

- **Niveau :**

  Le niveau de détail le plus bas est 1 (valeur par défaut) et le plus élevé est 9.

  Mais une trace de niveau 4 est très largement suffisante, puisque à ce niveau, toutes les valeurs des champs envoyées ou reçues sont détaillées.

- **Imprimer l'horloge pour chaque opération :**

  Activez l'écriture de l'horloge pour avoir une idée des temps de réponse entre chaque requête. Le niveau 1 suffit dans ce cas.

- **Membre :**

  Option facultative.

Téléchargement de la trace

Pour télécharger la trace AS/400 qui a été générée, il vous suffit de remplir les informations relatives à l'accès de la trace sur l'AS/400. Si la trace AS/400 est déjà active, ces informations sont pré-remplies.

- **Fichier destination :**

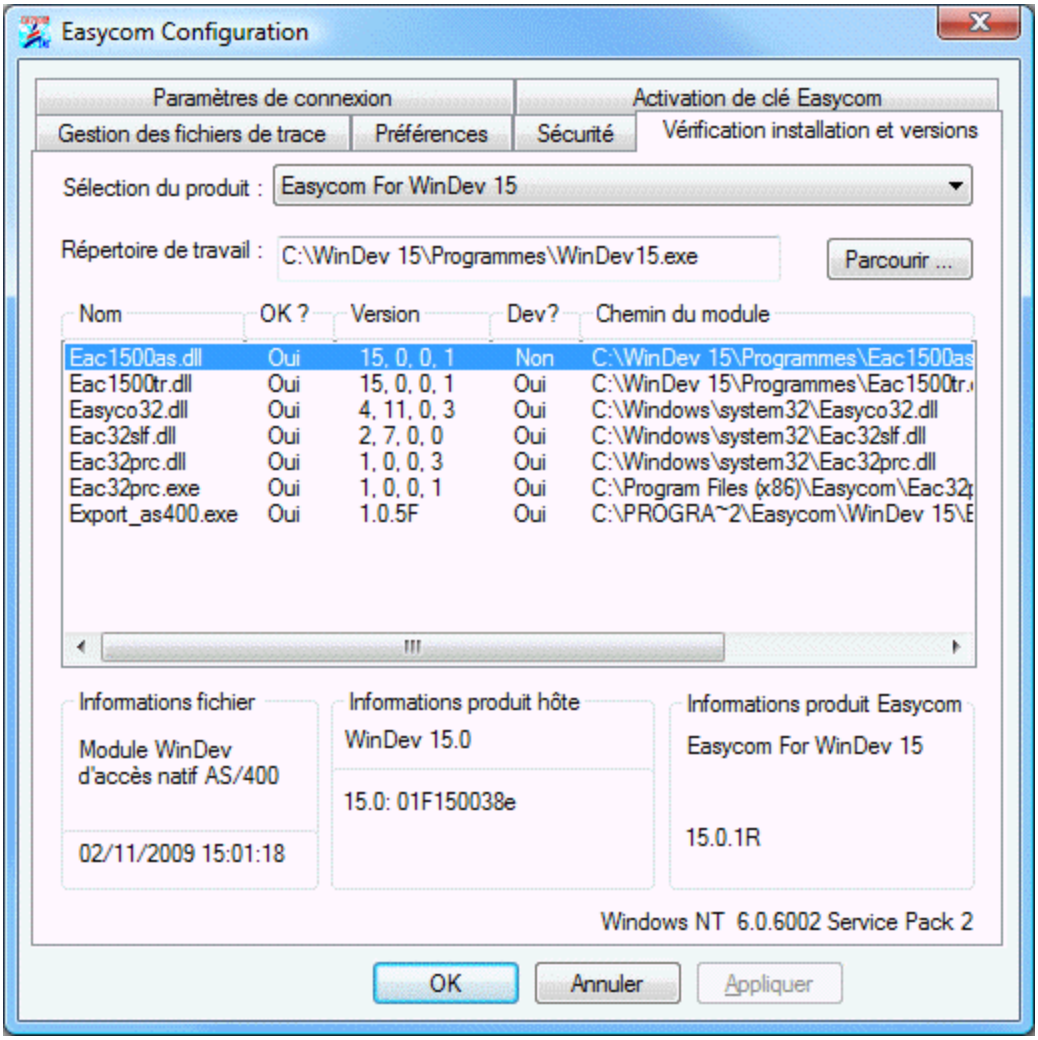
  Il vous reste à spécifier le fichier sur le PC qui sera généré en remplissant le champ de saisie ou en choisissant le fichier dans l'arborescence par le bouton **"..."** à droite de la zone de saisie.

  NB : Un nom utilisateur et un mot de passe doivent être renseignés sur l'onglet "Paramètres de connexion".

- **Télécharger :**

  Cliquez sur **"Télécharger"** pour commencer à récupérer la trace.

Versions des modules



Vous trouverez ici l'ensemble des dlls utilisées par Easycom et pour chaque dll la version du fichier et l'endroit par défaut où la dll est recherchée.

easycom.ini

Le fichier easycom.ini contient un ensemble de paramètres et d'options globales (installation, optimisation, trace, etc...). Notamment les paramètres choisis depuis l'utilitaire Easycom Configuration.

Il est possible d'avoir plusieurs easycom.ini, dans ce cas il sera recherché en premier dans le répertoire de l'application, puis dans le répertoire de Windows, puis dans les autres chemins définis dans le path.

Exemple de fichier Easycom.ini

```
[INSTALL]
PCdir=C:\PROGRAM FILES\Easycom

[GENERAL]
Network=TCP
Msg=1     //Option 'Afficher les messages'
NoWait=1
QryOptimize=a
Location=194.206.165.100  //Nom ou adresse IP de l'AS/400

[TCP]
COMPRESSION=0
Timeout=5    //Délai maximum de connexion TCP/IP

[Buffers]
Record=9    //Nombre maximum d'enregistrements récupérés en une
opération de lecture.
Size=8000    //Taille maxi en octet des données envoyées entre
l'AS/400 et le PC.
TimeOut=30    //Délai de rafraîchissement des données

[WINDEV]
INITLIBL=PROD2005;STATS2005
```

Annexes

Copyright

Contrat de Licence

CONTRAT DE LICENCE D'UTILISATION LOGICIEL " EASYCOM FOR WINDEV "

ATTENTION : Vous devez lire attentivement ce document avant d'utiliser le LOGICIEL dont vous venez d'acquérir un droit d'utilisation. En ouvrant l'emballage scellé contenant le CD Rom, vous vous engagez à respecter les termes de cette Licence. Si vous ou votre CLIENT, n'êtes pas d'accord avec les termes de cette LICENCE, vous devez retourner sous 24 Heures ce LOGICIEL complet à l'adresse où vous l'avez acheté pour obtenir son remboursement, SANS OUVRIR CETTE ENVELOPPE.

PROPRIETE DU LOGICIEL

1. Le LOGICIEL Easycom For WinDev (" LOGICIEL "), ainsi que la documentation produit (" Documentation ") sont et restent la propriété de AURA Equipements bénéficiant de la protection en matières de droit d'auteur et de droit des marques commerciales prévues par la loi française et les lois internationales en pareille matière.

DEFINITIONS

2. Le CLIENT: Le CLIENT représente la personne (physique ou morale ayant acquis la Licence ou utilisant une Version d'Evaluation).

3. Version d'évaluation: Vous utilisez une Version d'évaluation du LOGICIEL lorsque vous avez installé le LOGICIEL sans l'activation du numéro de série (clé d'activation). Après vingt minutes d'utilisation, un message s'affiche pour vous indiquer que vous utilisez une Version d'Evaluation. Il apparaîtra lors de l'utilisation du LOGICIEL toutes les minutes après une heure d'utilisation, et ceci dans la limite de 100 connexions pour une période de 15 jours après la date d'installation.

4. LOGICIEL: désigne le droit d'utilisation du LOGICIEL Easycom For WinDev acquis par le CLIENT.

(a) Installation: Le LOGICIEL Easycom For WinDev est composé d'une partie serveur qui s'installe sur une unité centrale AS/400 (Easycom For WinDev Serveur), et d'une partie cliente (Easycom For WinDev Client) qui s'installe sur les stations PC en liaison avec le Serveur.

(b) Le LOGICIEL serveur ne peut être installé qu'une seule fois sur une seule et même unité centrale AS/400 (N° de série de l'unité Centrale AS/400 unique). Si le LOGICIEL doit accéder à plusieurs sites différents, une licence du LOGICIEL doit être acquise pour chacun des sites.

(c) L'utilisation du LOGICIEL Client (licence de développement ou de déploiement) est fonction du nombre de sessions simultanées pour lequel vous avez acquis un droit de licence (la clé d'activation du LOGICIEL comprend le nombre de sessions AS/400).

(d) La Version d'Evaluation comprend un LOGICIEL Serveur et un LOGICIEL Client - 2 sessions.

4.1 Le LOGICIEL Easycom For WinDev de " DEVELOPPEMENT " permet le développement d'application. Le LOGICIEL comprend l'ensemble des fonctionnalités pour permettre le développement d'une application.

4.2 Le LOGICIEL Easycom For WinDev de " DEPLOIEMENT " permet l'exécution des développements effectués avec Easycom pour WinDev de " DEVELOPPEMENT ". Le LOGICIEL ne comprend pas l'ensemble des instructions nécessaires au développement.

4.3 La Version d'Evaluation est une version du LOGICIEL Easycom For WinDev de " DEVELOPPEMENT ".

5. Maintenance Mise à jour: comprend des corrections d'erreurs et ajout de fonctionnalités mineures ou fiabilisation de fonctions du LOGICIEL. Elle est identifiée par une modification du numéro à droite du point décimal (ex: identification de la nouvelle version du LOGICIEL " 10.1 " sera une Mise à jour de la Version 10 ou 10.0 du LOGICIEL).

6. Nouvelle Version: comprend des fonctions ou modules supplémentaires. Une nouvelle version est identifiée par une modification du numéro à gauche du point décimal (ex: une modification de " 10 " en " 11 ". AURA EQUIPEMENTS est le seul détenteur du droit de décider de la disponibilité des nouvelles Versions et/ou Mises à jour.

TERMES DE LA LICENCE

7. Licence pour Version d'évaluation: La version d'évaluation dans les limites et indications portées au paragraphe 3 est librement utilisable dans le seul but des tests du LOGICIEL en vue de l'acquisition d'une licence d'utilisation. La reproduction et distribution de la version d'évaluation (à l'exception expresse de la documentation)sont autorisées à condition de ne pas modifier les termes de cette licence, copyright, trademark, documentation et tout ou partie du LOGICIEL.

En aucun cas, le CLIENT ne pourra rechercher AURA EQUIPEMENTS en responsabilités pour des pertes ou dommages subis lors de la diffusion de la Version d'Evaluation.

8. Acquisition de la Licence d'utilisation: Si vous avez une Version d'Evaluation du LOGICIEL et que vous souhaitez bénéficier des droits d'utilisation du LOGICIEL décrits dans les sections 9,22 et 23, vous devez obtenir une clé d'activation, qui désinstalle les messages décrits dans la section 3. Vous devez acquérir une LICENCE d'utilisation pour obtenir une clé d'activation.

9. Licence: En vue de préserver et de protéger ses droits dans le cadre de la législation applicable, AURA EQUIPEMENTS ne vend pas de droit sur le LOGICIEL, mais octroie le droit d'exploiter ce LOGICIEL dans le cadre d'un accord de LICENCE et conserve expressément le droit de propriété de tous les LOGICIELS. Cette LICENCE d'utilisation non exclusive est concédée au CLIENT contre versement d'une redevance dépendant (i)du nombre de postes clients connectés à un seul serveur, et (ii)du critère de qualification du LOGICIEL soit " DEVELOPPEMENT " et/ou " DEPLOIEMENT ".

UTILISATION

10. Le CLIENT ne peut copier le LOGICIEL ou la documentation, excepté (i) comme autorisé dans la section 7 et (ii) pour faire une copie de sauvegarde à des fins de sauvegarde ou d'archive.

11. Il est interdit de modifier ou tenter de modifier ce LOGICIEL, de le traduire ou tenter de le traduire, de décompiler ou tenter de décompiler, de créer ou tenter de créer des travaux dérivatifs qui s'en inspireraient, de désassembler ou tenter de désassembler ce LOGICIEL.

12. Le CLIENT ne pourra, en aucun cas, louer, prêter, vendre, donner, divulguer ou mettre à disposition d'un tiers le LOGICIEL. Par exception, il pourra dans le cadre de l'article 7 distribuer gratuitement la version d'évaluation.

13. Le CLIENT ne pourra diffuser par le réseau Internet tout ou partie de la documentation ou du LOGICIEL excepté comme autorisé dans la section 7. Il est interdit de modifier ou tenter de modifier les clés d'activation (i)du LOGICIEL acquis par Licence d'utilisation (ii) de la Version d'Evaluation.

14. Le CLIENT ne bénéficie pas des droits de reproduction ou de distribution de tout ou partie du code source du LOGICIEL. La duplication de la documentation, en totalité ou en partie est formellement interdite.

15. Le CLIENT gardera confidentiel son code d'accès au LOGICIEL. Il s'engage à n'autoriser l'accès au LOGICIEL qu'à ses seuls salariés pour lesquels un tel accès est nécessaire.

16. Il est interdit d'enlever ou de tenter d'enlever les mentions de copyright pouvant apparaître et/ou étant contenues dans le LOGICIEL.

GARANTIE LIMITEE

17. Le LOGICIEL et la documentation qui l'accompagne sont vendus en l'état, sans aucune garantie d'aucune sorte.

18. AURA EQUIPEMENTS ne fournit aucune garantie expresse ou implicite, de quelque nature que ce soit sur le LOGICIEL et les services fournis. Notamment, le CLIENT ne saurait invoquer l'inaptitude du LOGICIEL et des services rendus à atteindre les objectifs qu'il se serait fixés.

19. Si le support physique LOGICIEL venait à être défectueux, le CLIENT se procurera auprès de AURA EQUIPEMENTS un nouveau support dans la limite d'une période de 30 jours à compter de la date d'acquisition.

20. Cette garantie ne sera pas appliquée si la détérioration provient d'un accident ou d'une mauvaise utilisation.

21. En aucun cas AURA EQUIPEMENTS ne pourra être tenu responsable de tout dommage direct, indirect, secondaire ou accessoires (y compris les dommages entraînés par la perte de bénéfices, l'interruption des activités ou la perte d'informations et autres) découlant de l'utilisation ou de l'impossibilité d'utilisation du produit, et ce même si AURA EQUIPEMENTS a été informé de la possibilité de dommages. La présente garantie limitée est régie par les lois en vigueur en France au bénéfice de AURA EQUIPEMENTS.

SERVICES DE MAINTENANCE

22. Support Technique: Vous devez compléter et retourner la carte d'enregistrement auprès de AURA Equipements. L'assistance n'est effective qu'après retour de ce Bon de Garantie chez AURA Equipements. Si vous avez acquis une licence d'utilisation du LOGICIEL VERSION DEVELOPPEMENT, vous avez droit pendant 30 jours à partir de la date de livraison de la clé d'activation du LOGICIEL à l'accès au support technique de AURA Equipements : contacts avec le service du support technique de AURA Equipements par fax, e-mail, forum Internet. Ce support consiste à répondre aux questions générales d'installation du LOGICIEL. Après cette période le support technique concernant ce LOGICIEL est assuré uniquement par l'intermédiaire du service payant " ASSISTANCE PREMIUM ".

23. Mise à jour et Version: AURA EQUIPEMENTS met à la disposition du CLIENT les mises à jour du LOGICIEL et les nouvelles versions de ce dernier correspondantes uniquement aux évolutions technique de logiciel EASYCOM, dans le cadre d' un contrat de " Mise à Jour " annuel souscrit auprès de AURA EQUIPEMENTS, au tarif en vigueur.

SONT EXCLUS DE CE SERVICE LES CHANGEMENTS DE VERSION INDUITS PAR UNE NOUVELLE VERSION DU LOGICIEL WINDEV LUI-MEME.

DIVERS

24. Validité: La licence d'utilisation du LOGICIEL démarre à la date de livraison de la Clé d'activation et continuera tant que le LOGICIEL sera protégé par copyright. S'il s'avère que le CLIENT ne respecte pas pour quelles raisons que ce soient les clauses du contrat, il devra immédiatement détruire l'ensemble du LOGICIEL et ses copies ainsi que la documentation et en apporter la preuve à AURA EQUIPEMENTS.

25. Juridiction: Ce contrat de LICENCE est régi par le droit français, tous litiges relatifs à la validité, à l'interprétation, l'exécution ou inexécution du présent contrat seront de la compétence exclusive des tribunaux du siège social de AURA EQUIPEMENTS.

AURA Equipements

129 rue de l'Abbé Groult, 75015 PARIS, France

TEL : +33 (0)1 53 76 86 35

www.easycom-aura.com

Support technique

Pour les demandes de support il faut créer un ticket via le site internet du support : http://support.easycom-aura.com/.
Il faut s'enregistrer la première fois, afin de recevoir un mot de passe.

Contrat d'assistance

AURA Equipements vous propose plusieurs niveaux de support technique.

Contactez-nous, nous vous ferons parvenir la meilleure offre commerciale répondant à vos besoins.

Pour des informations générales ou commerciales info@easycom-aura.com.

Commandes générales AS/400

Vous trouverez ci-dessous une liste de commandes qu'il est bon de connaître quand on travaille sur un AS/400. Certaines de ces commandes sont plus utiles à une personne s'occupant d'exploitation, d'autres sont plus intéressantes pour un programmeur.

Toutes ces commandes sont documentées sur l'AS/400. Pour afficher l'aide, tapez la commande, puis pressez la touche F4 afin de faire apparaître l'écran de saisie paramétré. Puis pressez la touche de fonction "F1" (ou la touche "Aide" équivalente) pour afficher l'aide correspondant à cette commande (prendre soin de placer le curseur au dessus du premier paramètre de la commande, sinon l'aide affichée correspondra au premier paramètre et non à la commande elle-même).

Général

DSPDBR	Afficher relations BD
DSPMSGD	Description du message
DSPSYSVAL	Pour afficher/modifier des variables du système. On y trouve notamment le numéro de série de l'AS/400 (QSRLNBR) et le codage des caractères par défaut (QCCSID)
DSPLIB	Afficher une bibliothèque
GO ASSIST	Menu d'assistance pour utilisateurs débutants
GO LICPGM	Permet de connaître la liste des programmes installés sur un AS/400
STRSQL	Démarrage de SQL/400
STRSEU	Démarrer l'éditeur de source SEU
STRPDM	"Start Program Manager", gestion des bibliothèques, objets et membres
SBMJOB	Soumettre un travail en file d'attente de travaux
SAVLIB/RSTLIB	Sauvegarder/Restaurer une bibliothèque
SAVOBJ/RSTOBJ	Sauvegarder/Restaurer un objet
WRKJOBSCDE	Gérer les postes du planning de travaux
WRKOBJPDM	Gestion des objets par PDM
WRKOUTQ	Gestion de toutes les files d'attente en sortie

WRKSYSSTS	Gérer l'état du système
WRKSPLF	Gestion de tous les fichiers spoule
WRKOBJLCK	Gestion des verrouillages d'un fichier (travaux, membres, enregistrement, …)
WRKTBL	Gestion des tables de caractères

Bibliothèques

ADDLIBLE	Ajouter une bibliothèque à la LIBL
CHGCURLIB	Changer la bibliothèque en cours

Fichiers

CHGPF	Changement des propriétés du fichier physique.
DSPPFM	Afficher le contenu d'un fichier physique
DSPFD	Afficher les propriétés d'un fichier
DSPFFD	Afficher les propriétés des zones d'un fichier
UPDDTA	Mise à jour de données

Travaux

WRKSBMJOB	Gestion des travaux soumis (Batch)
WRKACTJOB	Gestion des travaux actifs du système
WRKUSRJOB	Gestion des travaux d'utilisateur(s)
WRKJOBQ	Gestion de toutes les files d'attente de travaux
DSPJOBLOG	Afficher l'historique d'un travail
DSPJOBD	Afficher la description d'un job

Gestion des Transactions (Journalisation)

CRTJRN	Créer un journal
CRTRCVJRN	Créer le récepteur du journal
STRJRNPF	Démarrer le journal
DSPJRN	Afficher un journal

Liste des types d'objets AS400

Vous trouverez ci-dessous la liste complète des objets AS400 :

*ALRTBL
*CSPMAP
*AUTL
*CSPTBL
*BNDDIR
*CTLD

*CFGL
*DEVD
*CHTFMT
*DOC
*CLD
*DTAARA
*CLS
*DTADCT
*CMD
*DTAQ
*CNNL
*EDTD
*COSD
*EXITRG
*CRG
*FCT
*CRQD
*FILE
*CSI
*FLR
*FNTRSC
*JRN
*FNTTBL
*JRNRCV
*FORMDF
*LIB
*FTR
*LIND
*GSS
*LOCALE
*IGCDCT
*MEDDFN
*IGCSRT
*MENU
*IGCTBL
*MGTCOL
*IMGCLG
*MODD

*IPXD
*MODULE
*JOBD
*MSGF
*JOBQ
*MSGQ
*JOBSCD
*M36
*M36CFG
*PNLGRP
*NODL
*PRDAVL
*NTBD
*PRDDFN
*NWID
*PRDLOD
*NWSCFG
*PSFCFG
*NWSD
*QMFORM
*OUTQ
*QMQRY
*OVL
*QRYDFN
*PAGDFN
*SBSD
*PAGSEG
*SCHIDX
*PDFMAP
*SPADCT
*PDG
*SQLPKG
*PGM
*SQLUDT
*SQLXSR

*SRVPGM
*SSND
*S36
*TBL
*TIMZON
*USRIDX
*USRPRF
*USRQ
*USRSPC
*VLDL
*WSCST

Retour à la commande ASObjLocks.

Foire aux Questions FAQ

Comment vérifier les numéros de version ?

Utiliser Easycom Configuration :

- pour la partie serveur faire un test de connexion depuis l'onglet "Paramètres de connexion",
- pour la partie cliente (dlls) dans l'onglet "Vérification installation et versions"

On peut aussi consulter la version serveur par la commande `CALL EASYCOM/EASYCOM *VERSION.`

La commande `WRKLICINF` donne la version OS/400.

Comment modifier le nom du sous-système ?

Il ne peut y avoir qu'un sous-système par bibliothèque Easycom, pour avoir plusieurs sous-systèmes, dupliquer la bibliothèque EASYCOM et lancer la commande CFGEACSBS pour spécifier le nom du nouveau sous-système.

Dans ce cas il faut bien sûr configurer les postes clients avec le nom du nouveau sous-système.

Comment changer le port d'Easycom Serveur ?

Par défaut Easycom Serveur utilise le port 6077, vous pouvez le modifier par la commande CFGEACSBS de la bibliothèque EASYCOM.

Dans ce cas, spécifier le nouveau port à la suite de l'adresse IP dans les infos étendues ou la propriété Serveur de la connexion (Easycom Configuration)

194.160.205.192:6079

Faut-il ajouter la bibliothèque Easycom dans les JOBD des utilisateurs ?

Non. La bibliothèque s'ajoute automatiquement à l'initialisation du job et n'a pas besoin de se trouver dans la LIBL de chaque profil.

Comment ajouter une bibliothèque ?

Plusieurs solutions sont possibles :

- Globalement et pour toutes les connexions, dans le fichier easycom.ini, ajouter INITLIBL=BIBLIO1;BIBLIO2 dans la section [WINDEV]
- Au niveau des infos étendues de la connexion, dans l'option INITLIBL,
- Au niveau des propriétés d'EACJOBD,
- En activant l'option de sécurité du serveur Easycom, le programme EACP003 peut ajouter des bibliothèques,
- Par l'envoi d'une commande `AsExec("ADDLIBL BIBLIO")`

Comment ouvrir un fichier qui n'est pas dans l'analyse ?

Il faut utiliser la fonction HDeclareExterne.

Comment changer de membre ?

Le plus simple est d'utiliser la fonction ASPropriete :

```
ASPropriete(FICHIER, "MEMBER", "nom_du_membre")
```

Un membre peut également être spécifié dans les infos étendues du fichier :

MAIN=BIBLIO/FICHIER(MEMBRE) ou

MAIN=

```
1)Il faut modifier les 'infos etendues' dans la description du fichier :
<EASYCOM>
MAIN=ERVFLVA/STOCK(LVA_BOUTIQ)
</EASYCOM>

2)Il faut modifier les 'infos etendues' dans la description des rubriques :
<EASYCOM>
NATIVETYPE=0
LF=ERVFLVA/STOCK(LVA_BOUTIQ)
</EASYCOM>

3)Il faut ensuite spécifier dans le programme d'initialisation :
ASPropriete(Stock, "MEMBER", "LVA_BOUTIQ")
```

Peut-on travailler en même temps avec des fichiers Hyper File et AS/400 ?

Dans un même projet, vous pouvez travailler à la fois avec des fichiers Hyper File et des fichiers AS/400.

Il suffit de définir des connexions différentes dans l'analyse (ou dans le code) : une connexion de type Accès Natif AS400 et une connexion Hyper File (ou autre).

Ouvrir ou changer la connexion **avant** l'ouverture du fichier.

Peut-on modifier les noms des rubriques ?

Oui, d'autres noms de champs que ceux définis dans le fichier physique de l'AS/400 peuvent être utilisés. C'est grâce aux 'infos étendues' du fichier et des 'infos étendues' des rubriques que le fichier AS/400 sera toujours reconnu.

Peut-on supprimer des rubriques d'un fichier dans l'analyse ?

Oui, il est possible de supprimer des rubriques (dans la description du fichier de l'analyse) mais il ne sera alors pas possible de faire des ajouts d'enregistrements sur le fichier.

Comment gérer plusieurs bibliothèques ?

Si le nom de la bibliothèque n'est pas explicitement défini dans les infos étendues de la connexion ou du fichier, il sera recherché en fonction de la LIBLE du JOB.

Par défaut c'est celle du profil mais elle peut être définie ou complétée par les propriétés options et les programmes de sécurité.

```
ASExec("ADDLIBLE LIBRARY")
ASExec("RMVLIBLE LIBRARY")
```

Comment gérer plusieurs AS/400 ?

Il suffit d'avoir une connexion par AS400, soit dans l'analyse, soit définies en programmation.

On peut établir explicitement une connexion à un AS/400 particulier par la fonction HOuvreConnexion.

On peut associer un ou plusieurs fichiers à une connexion par la fonction HChangeConnexion.

Note : pour accéder à plusieurs AS/400, vous devez disposer d'une licence par AS/400.

Comment connaître le nombre d'enregistrements d'un fichier ?

Le plus simple est d'utiliser la fonction HNbEnr.

Les options (enregistrements actifs, rayés ou supprimés) sont ignorées, seul le nombre d'enregistrement total est retourné.

Pour les autres fichiers vous pouvez utiliser l'opérateur COUNT dans une requête SQL :

```
MaRequete = "SELECT COUNT(CUST_ID) AS NOMBRE FROM BIBLIO/FICHIER"
HExécuteRequête(MaRequete,MaConnexionas400)
```

Comment ne pas afficher les boites de dialogue Easycom ?

Depuis le groupe de programmes, lancez "Easycom configuration", dans l'onglet "Optimisations d'Easycom" décocher l'option "Afficher les messages".

Il faut dans ce cas gérer toutes les erreurs éventuelles en programmation.

Il est également possible d'utiliser l'option **SHOWDIALOGS=0** dans les infos étendues de la connexion.

Comment lancer une application Windev depuis un programme AS400 ?

Si vous avez Client Access, vous pouvez utiliser la commande STRPCPGM.

Aura Equipements édite également Launcher 400 qui permet de piloter des programmes PC mais également de concevoir des documents Office (Word, Excel) ou d'accéder aux bases de données SQL.