4D Internet Commands

Les commandes Internet de 4D

Préface

Les commandes Internet de 4D (“4D Internet Commands”) offrent aux utilisateurs de 4D un ensemble puissant d'outils de communication permettant de travailler sur tout type de réseau, local, national ou mondial. Ces dernières années, le nombre d'utilisateurs et d'entreprises accédant à Internet s'est considérablement accru. Avec la croissance du nombre d'internautes, le besoin d'être présent "sur le net" est plus fortement ressenti chaque jour par les professionnels.

Les commandes Internet de 4D fournissent aux développeurs de bases de données 4D un accès à de nombreux éléments-clés d'Internet. Les commandes SMTP contiennent des outils permettant d'automatiser l'envoi de messages électroniques depuis une base de données vers une liste illimitée de destinataires. De même, les commandes POP3 et IMAP permettent de récupérer du courrier d'un nombre illimité de boîtes aux lettres électroniques pour le stocker dans une base de données, le réacheminer, lui apporter une réponse automatique ou effectuer une recherche à distance. Les commandes FTP permettent de transférer des fichiers depuis/vers des systèmes distants ou encore de lister les documents et répertoires présents sur un volume FTP. Enfin, les commandes TCP et UDP fournissent aux développeurs des outils de bas niveau pour leur permettre d'exécuter de multiples tâches liées à Internet.

Le protocole SMTP (Simple Mail Transfer Protocol) est le principal protocole de transfert de courrier utilisé sur Internet. Les commandes Internet de 4D permettent de créer et d'envoyer rapidement du courrier via un serveur SMTP. La création et l'envoi de courrier peuvent s'effectuer à l'aide d'une seule commande. Si vos besoins en matière de messagerie sont plus complexes, les commandes Internet de 4D fournissent des outils permettant de contrôler entièrement les en-têtes, le corps et les pièces jointes des messages. Comme le courrier Internet peut être adressé à des réseaux "privés" tels que CompuServe, America Online, eWorld, etc., vous êtes donc en mesure d'atteindre virtuellement quiconque possédant un compte e-mail. Le jeu de commandes SMTP permet également, par exemple :

- l'envoi automatisé de statistiques et d'états depuis vos bases de données ;
- la création d'une base de données de réacheminement automatique de courrier ;
- la gestion d'une liste de diffusion de groupe (mailing list) ;
- la synchronisation de bases de données distantes.

Outre les commandes SMTP, les commandes Internet de 4D contiennent également des outils permettant la connexion à des serveurs de courrier électronique POP3 (Post Office Protocol, Version 3) ou IMAP (Internet Message Access Protocol) pour récupérer des messages électroniques et des documents joints encodés. Les commandes SMTP, POP3 et IMAP sont conformes aux normes MIME s'appliquant aux pièces jointes multiples, ce qui en facilite le téléchargement et la sauvegarde.

Des commandes permettent également aux utilisateurs d'encoder les documents joints sous divers formats tels que Binhex, Base64, AppleSingle, AppleDouble...

Les commandes FTP (File Transfer Protocol) sont très simples d'emploi pour envoyer des fichiers texte ou binaires à un serveur FTP et en recevoir. Les commandes FTP permettent d'obtenir la liste des répertoires et des fichiers, facilitant ainsi le développement d'interfaces de navigation vers des volumes distants. On peut facilement intégrer des commandes FTP dans des applications de recherche de documents sans devoir “monter” les volumes distants sur le poste client.

Le protocole TCP/IP (Transmission Control Protocol/Internet Protocol) est le principal protocole utilisé pour l'envoi et la réception de données sur Internet. Plusieurs commandes Internet de 4D permettent d'envoyer et de recevoir des paquets TCP. Les commandes TCP fournissent aux développeurs les outils nécessaires à la construction et au contrôle de leurs communications sur Internet. La commande *TCP_Open* autorise notamment la connexion à un serveur en mode sécurisé à l'aide du protocole TLS (Transport Layer Security).

Par exemple, vous pourrez ainsi :

- construire votre propre interface telnet ;
- exécuter des instructions sur des machines distantes ;
- récupérer des documents sur le World Wide Web ;
- effectuer des recherches dans de multiples bases de données en ligne ;
- gérer la synchronisation de bases de données avec des serveurs distants ;
- suivre des colis confiés à Federal Express ou UPS ;
- vous connecter à un serveur Web en HTTPS.

Note : Pour une plus grande souplesse, les commandes Internet de 4D permettent de passer directement une référence de connexion POP3, IMAP ou FTP aux commandes TCP de bas niveau et inversement. Pour plus d'informations, reportez-vous à la section **Routines de bas niveau, Présentation**.

Le protocole UDP (User Datagram Protocol) est un protocole non connecté permettant l'envoi et la réception de données de façon plus simple et plus rapide que TCP mais avec un niveau de fiabilité moindre. Il est utilisé lorsque le besoin de rapidité est élevé, par exemple dans le cas de la transmission de flux de données (streaming). Les commandes UDP fournissent les outils essentiels pour envoyer et recevoir des données via ce protocole.

Installation et logiciels requis

Installation

Les commandes Internet de 4D sont installées dans 4D par l'intermédiaire d'un plug-in, appelé “4D Internet Commands”.

4D Internet Commands est disponible dès que vous installez 4D ou 4D Server. Il est copié dans le dossier **PlugIns** de votre application.

Pour plus d'informations sur l'installation et la configuration des plug-ins, veuillez consulter le Guide d'installation de 4D Product Line.

Configuration requise

La configuration système requise pour les commandes Internet de 4D est identique à celle de 4D. Pour plus d'informations, reportez-vous au *Guide d'installation* de 4D.

Versions 64 bits

4D Internet Commands version 64 bits doit être utilisé avec 4D Server 64 bits :

- Windows: 4D Server 64 bits disponible à partir de la version 12.1
- OS X: 4D Server 64 bits pour OS X disponible à partir de la version 14 R3

Accès au réseau

Pour pouvoir utiliser les commandes Internet de 4D, vous devez avoir accès à un réseau exploitant le protocole TCP/IP.

TLS

4D Internet Commands permet d'utiliser le protocole sécurisé avec les commandes d'envoi de messages et de connexion aux serveurs de messagerie. L'utilisation de ce protocole ne nécessite aucune configuration particulière.

Note : L'implémentation de TLS dans 4D Internet Commands utilise la "méthode implicite".

Serveur de noms de domaine

Pour de nombreuses commandes Internet de 4D, il est nécessaire d'avoir accès à un serveur de noms de domaines (Domain Name Server, ou DNS).

Serveur de courrier électronique SMTP

Pour envoyer du courrier au moyen des commandes SMTP, l'expéditeur doit avoir accès à un serveur de courrier électronique SMTP.

Serveur de courrier électronique POP3

Pour utiliser les commandes POP3, vous devez avoir un compte sur un serveur de courrier électronique POP3.

Serveur de courrier électronique IMAP

Pour utiliser les commandes IMAP, vous devez avoir un compte sur un serveur de courrier électronique IMAP.

Glossaire et terminologie

Cette partie définit succintement la plupart des concepts et termes spécifiques utilisés dans ce manuel. La section suivante, "Format des paramètres", fournit des informations supplémentaires sur les paramètres des commandes Internet de 4D.

NIC : "Network Information Center" (Centre d'informations sur le réseau). Internet est, de manière générale, une entité non réglementée. Il n'existe pas d'autorité centralisant ou contrôlant son utilisation et son développement. Toutefois, il est nécessaire qu'un organisme unique régule les attributions de noms de domaine et d'adresses IP. Le NIC remplit ce rôle.

RFC : "Request for Comments". La plupart des commandes Internet de 4D s'appuient sur des normes définies pour traiter les communications sur Internet. Les méthodologies, descriptions et protocoles standard utilisés sur Internet sont définis dans des documents intitulés RFC. L'**Annexe D, Informations supplémentaires...** référence les sites Web comportant des liens vers la plupart des documents RFC. Bien que les commandes Internet de 4D simplifient la programmation des accès à Internet, il peut parfois vous être utile de consulter certains de ces documents, en particulier si vous souhaitez utiliser les routines de communication TCP de bas niveau.

Adresses TCP/IP, noms de serveur et noms de domaine : L'adresse IP est la référence d'une machine **spécifique** se trouvant quelque part dans le monde. Une adresse IP se présente sous la forme d'une chaîne de caractères contenant quatre valeurs numériques séparées par des points (par exemple, "207.94.15.3"). Chaque partie numérique de l'adresse peut contenir une valeur comprise entre zéro et 255. En appliquant certaines fonctions mathématiques à une adresse IP, sa valeur peut être exprimée sous forme d'entier long (appelée **ip_EntierLong** dans ce manuel).

Afin que les ordinateurs d'un site (par exemple une entreprise, une université, etc.) puissent se connecter à Internet, des garanties doivent être prises pour s'assurer que leurs adresses IP ne rentrent pas en conflit avec d'autres machines sur le réseau. Les sociétés (et parfois les particuliers) peuvent donc inscrire leur site auprès du **NIC** afin d'obtenir un **nom de domaine**. L'utilisation de **noms de domaines** (par exemple "www.4d.com" ou "ftp.4d.com") facilite l'identification et la lecture des adresses Internet. Les noms de domaines sont traduits par le DNS (cf. ci-dessous) en adresses numériques (numéros IP) utilisées par le réseau.

Nom de domaine = "4d.com"

Nom d'hôte (Nom de serveur)	=	**Adresse IP**	=	**ip_EntierLong**
"www.4d.com"	=	"207.94.15.3"	=	-815919357

La correspondance entre un **Nom d'hôte** et son adresse IP est stockée dans une base de données appelée DNS (Domain Name System ou Système de noms de domaine). Les DNS communiquent entre eux pour échanger toute donnée nouvelle ou modifiée dans les listes de noms de domaines du monde entier. Le panneau de contrôle TCP/IP permet de “faire pointer” votre ordinateur sur un DNS, qui se chargera alors de traduire les références des noms de domaine que vous utilisez.

Il est important de noter que tous les serveurs de noms de domaines disposent d'une adresse IP. Cependant, toutes les adresses IP n'ont pas un serveur de noms de domaine correspondant. De même, une adresse électronique telle que "jsmith@4d.com" ne désigne pas l'ordinateur spécifique ou l'adresse IP de cette personne. L'adresse électronique dirige la distribution du courrier vers la machine dont l'adresse IP est obtenue en convertissant le domaine "4d.com". Le courrier est distribué au serveur POP3 fonctionnant sur cette machine, qui le met alors en attente pour l'utilisateur nommé "jsmith".

Nom de domaine : Le nom de domaine est une structure d'adressage utilisée pour l'identification et la localisation des ordinateurs sur Internet. Les noms de domaines facilitent la mémorisation des adresses Internet, qui sont traduites par le système de noms de domaines (DNS) en adresses numériques (numéros Internet Protocol [IP]) utilisées par le réseau. Un nom de domaine est structuré et fournit souvent des informations sur le type d'entité qu'il représente. Les noms de domaines situés au même niveau hiérarchique doivent être uniques. Par exemple, il ne peut y avoir qu'un seul *com* au niveau supérieur de la hiérarchie et un seul *4d.com* au niveau suivant de la hiérarchie. Si le nom de votre entreprise est "NomEntreprise", vous pourriez enregistrer le nom de domaine "NomEntreprise.com" et votre adresse de courrier électronique pourrait être "NomUtilisateur@NomEntreprise.com". Vos clients pourraient accéder à votre site Web en se connectant à "www.NomEntreprise.com" avec leur navigateur Web.

Système de noms de domaine (DNS) : Il s'agit d'une base de données répartie contenant des informations utilisées pour traduire des noms de domaines, qui sont faciles à mémoriser et à utiliser, en numéros Internet Protocol (IP), nécessaires pour localiser les ordinateurs sur Internet. Les utilisateurs du monde entier conservent et mettent jour la partie de la base de données qui les concerne, et la totalité de cette base est disponible pour tous les ordinateurs et utilisateurs d'Internet. Le DNS comprend les ordinateurs, les fichiers de données, les logiciels et les personnes travaillant ensemble.

Encodage : L'encodage est utilisé pour convertir un fichier dans un format interprétable par tout type de système d'exploitation (ASCII standard). Le type d'encodage le plus répandu est l'encodage binaire-héxadécimal (Binhex). C'est l'option d'encodage par défaut de la plupart des documents que vous joignez à vos messages. Un fichier encodé est plus gros que l'original.
Sous Mac OS, l'encodage permet de convertir la partie "données" (data fork) et la partie "ressources" (resource fork) d'un fichier en un document de type texte qui peut être facilement envoyé comme document joint. Les commandes Internet de 4D acceptent les modes d'encodage les plus courants dont Binhex, Base64, AppleSingle, AppleDouble, UUEncode et MacBinary.

Encryptage : L'encryptage est utilisé pour brouiller intentionnellement le contenu des messages. Les messages sont codés au moyen d'un programme externe de cryptage, tel que PGP, dans le seul but d'accroître leur confidentialité. Le texte encrypté doit alors être décrypté avant d'être lu. Les commandes Internet de 4D **ne fournissent pas** de moyen d'encrypter des messages.

Compression : La compression est utilisée pour réduire l'espace disque occupé par un fichier. Pour compresser un fichier, vous pouvez utiliser une application telle que Stuffit Deluxe™ Compact Pro™ ou WinZip™. Le fichier doit ensuite être décompressé par l'application qui l'a créé pour retrouver son format original. Les applications de compression ajoutent généralement un suffixe au nom original du fichier. Voici quelques suffixes courants et leurs applications respectives :

Nomfichier.SIT - application Stuffit
Nomfichier.CPT - application Compact Pro
Nomfichier.DD - application Disk Doubler
Nomfichier.ZIP - application Winzip
Nomfichier.SEA - Self Extracting Archive. Ces fichiers sont des applications Macintosh auto-extractibles, ils se décompressent seuls lorsque l'utilisateur double-clique dessus car le code de décompression est inclus. En raison de l'ajout de ce code, les archives auto-extractibles sont généralement plus volumineuses que les autres. Cependant, puisque l'utilisateur n'a pas besoin de l'application de compression, cette option peut s'avérer avantageuse.

Il est important de noter qu'une fois compressé, un fichier doit être encodé afin de pouvoir être transmis correctement de machine en machine jusqu'à sa destination finale.

Format des paramètres

Cette section indique la signification et le formatage des paramètres clés utilisés dans ce manuel.

Paramètre	Type		Description
nomServeur	Texte	→	Nom du serveur (Ex. : "www.nomdesociete.com") ou Adresse IP (Ex. : "204.118.90.2")
ip_EntierLong	Entier long	→	Référence d'une adresse IP sous forme d'entier long
adresseEmail	Texte	→	Ex: "jsmith@4d.com"
listeAdresses	Texte	→	Ex: "jsmith@4d.com, jdupont@4d.fr" ou "jsmith@4d.com"+Caractere(13)+"jdupont@4d.fr"
cheminLocal	Texte	→	- Document Mac : "Disque dur:BDD:Ventes:Rapport" Win : "C:\Dossier\BDD\Ventes\Rapport.txt" - Dossier Mac : "Disque dur:En cours:" (Notez le ":" final) Win : "C:\EnCours\" (Notez le "\" final)
cheminServeur	Texte	→	- Document : "/usr/jsmith/rapports/rapportventes.txt" - Dossier : "/usr/jsmith/rapports/"(Notez le "/" final)
tcp_ID	Entier long	→	Référence d'une session TCP ouverte
smtp_ID	Entier long	→	Référence d'un nouveau message électronique
pop3_ID	Entier long	→	Référence d'une session POP3 ouverte
imap_ID	Entier long	→	Référence d'une session IMAP ouverte
ftp_ID	Entier long	→	Référence d'une session FTP ouverte
udp_ID	Entier long	→	Référence d'une session UDP ouverte
Résultat	Entier long	←	Code d'erreur

nomServeur

Le paramètre *nomServeur* est le nom ou l'adresse IP du serveur hôte (HostName), par exemple "dns.4d.com" ou "204.118.90.2". Les noms de serveurs sont convertis au moyen du système de noms de domaines. Les noms de domaines "par défaut" (primaires) et "secondaires" sont généralement indiqués dans le tableau de bord du gestionnaire TCP/IP installé. Les commandes Internet de 4D nécessitant un nom de serveur comme paramètre acceptent indifféremment son nom ("www.4d.com") ou son adresse IP ("204.118.90.2"). Le format "nom" est généralement préférable car il met l'application à l'abri d'effets indésirables liés aux modifications matérielles dans les sites distants.

ip_EntierLong

Des formules mathématiques peuvent être appliquées aux adresses IP pour les convertir en entiers longs uniques. Les commandes **NET_NameToAddr** and **NET_AddrToName** (thème **IC Internet**) automatisent cette conversion. Cet entier long est désigné par *ip_EntierLong* dans ce manuel. Cette valeur n'est utilisée que dans des circonstances particulières par des développeurs établissant des communication TCP directes.

Certains développeurs préféreront également stocker des entiers longs plutôt que des noms de domaines afin d'économiser de l'espace disque. Toutefois, pour des raisons de compatibilité avec IPV6, cette astuce est déconseillée. Il est préférable de travailler avec l'adresse IP ou le nom de domaine.

mailAddress

Le paramètre *adresseEmail* est une spécification d'adresse électronique complète au format "nom_utilisateur@nom_domaine". Dans ce manuel, *adresseEmail* fait référence à une adresse électronique unique. Lorsqu'un paramètre d'une routine accepte une liste de plusieurs adresses, le paramètre *listeAdresses* est explicitement indiqué.

adresseEmail ne peut accepter qu'une seule adresse électronique. Il doit comporter à la fois le nom de l'utilisateur et le nom de domaine :

"Felix Unger" <felix@pristine.com>
oscar@slobs.com (Oscar Madison)

addressList

Le paramètre *listeAdresses* contient une ou plusieurs adresses électroniques au format décrit dans le paramètre *adresseEmail*, les adresses étant séparées par des virgules ou des retours chariot. La délimitation par retour chariot est utile pour fournir aux utilisateurs une zone de texte permettant de saisir ou de coller plusieurs adresses. Les trois exemples suivants génèrent une valeur *$listeAdresses* valide:

```
$ListeAdresses:="jsmith@4d.com"
$ListeAdresses:="jsmith@4d.com,scott@4d.com,marcel@4d.fr"
For($i;1;Size of array(aAdresses))
   $ListeAdresses:=$ListeAdresses+aAdresses{$i}+Char(13)
End for
```

cheminLocal

Le paramètre *cheminLocal* indique l'emplacement d'un fichier ou d'un répertoire sur l'ordinateur de l'utilisateur (Mac ou Windows).

Sur un Macintosh, les éléments à l'intérieur des dossiers sont séparés par des caractères "deux-points" (:). Par exemple, le fichier "Mon rapport" dans le dossier "Rapports" sur le disque dur "Mon disque dur" aura comme chemin d'accès "Mon disque dur:Rapports:Mon Rapport". Une spécification de répertoire sur un Macintosh doit se terminer par un caractère "deux-points". Par exemple, si vous voulez placer un nouveau rapport dans le dossier indiqué précédemment, vous devez passer la chaîne "Mon disque dur:Rapports:". La décision de faire référence à un nom de fichier ou de répertoire est liée au contexte de la commande.

Sous Windows, le principe utilisé est identique, à l'exception du fait qu'une barre oblique inversée "\" est utilisée à la place des caractères "deux-points".

Note : Avec le protocole FTP, les noms des fichiers manipulés par les commandes Internet de 4D ont une taille limitée. Pour plus d'informations reportez-vous à la section **Transfert de fichiers, Présentation**.

cheminServeur

Le *cheminServeur* est l'emplacement d'un fichier ou d'un répertoire sur un ordinateur fonctionnant sous le système d'exploitation Unix. Dans l'environnement Unix, les répertoires sont séparés par des barres obliques ("/"). Par exemple, le fichier "rapport.txt" dans le répertoire "rapports" du répertoire "4D" sera désigné par "/4D/rapports/rapport.txt". Le chemin d'accès d'un répertoire doit se terminer par un caractère "/". Notez qu'un chemin d'accès complet commence par une "/" qui représente la racine du volume.

Note : Avec le protocole FTP, les noms des fichiers manipulés par les commandes Internet de 4D ont une taille limitée. Pour plus d'informations reportez-vous à la section **Transfert de fichiers, Présentation**.

smtp_ID, pop3_ID, imap_ID, ftp_ID, tcp_ID

La plupart des commandes Internet de 4D, quel que soit leur thème, utilisent un numéro d'ID permettant de référencer de façon unique leur session de travail. Toutes les commandes qui doivent s'exécuter dans le cadre d'une session y font référence par l'intermédiaire de l'ID de la session.

Les numéros d'"ID" sont propres à chaque classe de commande (SMTP, POP3, IMAP, FTP, TCP, UDP). Toutefois, il est possible de passer directement une référence de connexion POP3, IMAP ou FTP aux commandes TCP de bas niveau et inversement. Pour plus d'informations, reportez-vous à la section **Routines de bas niveau, Présentation**.

Référence de session	Ouverte par	Fermée par
tcp_ID	TCP_Open ou TCP_Listen	TCP_Close
smtp_ID	SMTP_New	SMTP_Clear
pop3_ID	POP3_Login	POP3_Logout ou POP3_VerifyID
imap_ID	IMAP_Login	IMAP_Logout ou IMAP_VerifyID
ftp_ID	FTP_Login	FTP_Logout ou FTP_VerifyID
udp_ID	UDP_New	UDP_Delete

Résultat

Toutes les commandes Internet de 4D (à l'exception de **IT_ErrorText** et **IT_Version**) retournent un entier comme résultat de la fonction. Cet entier contient un numéro d'erreur que la commande doit retourner à la base de données 4D. Si une commande aboutit, un zéro est renvoyé. Sinon, un code d'erreur est renvoyé. L'**Annexe C, Codes d'erreurs de 4D Internet Commands** fournit la liste des codes d'erreurs des commandes Internet de 4D.

IC Downloaded Mail

- Téléchargement de courrier, présentation
- MSG_Charset
- MSG_Delete
- MSG_Extract
- MSG_FindHeader
- MSG_GetBody
- MSG_GetHeaders
- MSG_GetMessage
- MSG_GetPrefs
- MSG_HasAttach
- MSG_MessageSize
- MSG_SetPrefs

Téléchargement de courrier, présentation

Les commandes préfixées "MSG_" permettent de manipuler des messages qui ont été enregistrés sous forme de fichiers locaux au moyen des commandes *POP3_Download* (thème **IC POP3 Review Mail**) ou *IMAP_Download* (thème **IC IMAP Review Mail**). Les commandes Internet de 4D étant entièrement compatibles MIME, vous pourrez aisément extraire et/ou décoder les fichiers joints. Pour plus d'informations sur les normes MIME, veuillez consulter les documents RFC 1521, RFC 1522 et RFC 2045.

Une fois les messages téléchargés et enregistrés localement, les commandes de ce thème fournissent différentes fonctions permettant de manipuler ces documents. Vous pouvez ainsi obtenir des informations sur les composantes du message, séparer l'en-tête du corps du message, détecter et extraire les fichiers joints, ou encore supprimer des documents.

MSG_Charset

MSG_Charset (décoderEntêtes ; jeuCorps) -> Résultat

Paramètre	Type		Description
décoderEntêtes	Entier	➡	-1 = Utiliser le paramétrage courant, 0 = Ne rien faire, 1 = Convertir dans le jeu de caractères Mac OS si ISO-8859-1 ou ISO-2022-JP, décoder les caractères étendus
jeuCorps	Entier	➡	-1 = Utiliser le paramétrage courant, 0 = Ne rien faire, 1 = Convertir dans le jeu de caractères Mac OS si ISO-8859-1 ou ISO-2022-JP
Résultat	Entier	↩	Code d'erreur

Description

La commande *MSG_Charset* automatise le traitement des caractères étendus dans les messages lors de leur exploitation via certaines commandes MSG. Si cette commande n'est pas appelée ou si ses deux paramètres sont mis à 0, les commandes Internet de 4D version 6.8.1 ou supérieure fonctionneront de la même manière qu'en version 6.5.x.

La commande *MSG_Charset* permet de définir, d'une part, si les en-têtes comportant des caractères étendus doivent être décodés et, d'autre part, si le jeu de caractères utilisé dans le corps des messages et dans les en-têtes doit être converti.

Cette commande est particulièrement utile pour le traitement des caractères étendus dans les en-têtes tels que "Subject" et les noms placés dans les adresses (par exemple, pour le décodage d'adresses sous la forme =?ISO-8859-1?Q?Test=E9?= <test@n.net>).

Le paramètre *décoderEntêtes* définit les traitements à appliquer aux champs d'en-tête lors de l'exécution de la commande *MSG_FindHeader*. Par défaut, ce paramètre a pour valeur 0.

- -1 : Utiliser les paramétrages courants ;
- 0 : Ne rien faire ;
- 1 : L'en-tête est décodé si nécessaire. Si l'en-tête est décodé et si le jeu de caractères spécifié est de l'ISO-8859-1 ou de l'ISO-2022-JP, il est converti, respectivement en ASCII Mac OS ou en Shift-JIS.

Le paramètre *jeuCorps* définit les traitements à appliquer au corps du message lors de l'exécution de la commande *MSG_GetBody*. Par défaut, ce paramètre a pour valeur 0.

- -1 : Utiliser les paramétrages courants ;
- 0 : Ne rien faire ;
- 1 : Si le jeu de caractères spécifié dans le champ "Body-Content-Type" est de l'ISO-8859-1 ou de l'ISO-2022-JP, le texte du corps du message est converti, respectivement en ASCII Mac OS ou en Shift-JIS.

Note de compatibilité (version 6.8.1) : Si la commande *MSG_Charset* n'est pas utilisée et que la commande *POP3_Charset* a été utilisée, les commandes *MSG_FindHeader* et *MSG_GetBody* prendront en compte les paramétrages de *POP3_Charset*. Si *MSG_Charset* est utilisée, les paramétrages de *POP3_Charset* sont ignorés.

Exemple

En utilisant une version 6.8.1 ou ultérieure de 4D Internet Commands :

```
$Err:=MSG_Charset(1;1)
$Err:=MSG_FindHeader($msgfile;"From";$from)
$Err:=MSG_FindHeader($msgfile;"To";$to)
$Err:=MSG_FindHeader($msgfile;"Cc";$cc)
$Err:=MSG_FindHeader($msgfile;"Subject";$subject)
$Err:=MSG_MessageSize($msgfile;$HdrSize;$BdySize;$msgSize)
$Err:=MSG_GetBody($msgfile;0;$BdySize;$BodyContent).
```

MSG_Delete

MSG_Delete (nomFichier ; dossier) -> Résultat

Paramètre	Type		Description
nomFichier	Texte	→	Nom ou chemin d'accès de fichier
dossier	Entier	→	0 = Dossier Messages, 1 = Dossier DocsJoints
Résultat	Entier	←	Code d'erreur

Description

La commande *MSG_Delete* supprime un fichier local.

Le paramètre *nomFichier* indique le nom ou le chemin d'accès complet du fichier à supprimer.
Le paramètre *dossier* est utilisé lorsque *nomFichier* contient un nom de fichier seul. Il permet de désigner le dossier dans lequel la suppression doit être effectuée :

- 0 : dossier des messages (défini par *POP3_SetPrefs* ou *MSG_SetPrefs*),
- 1 : dossier des documents joints (défini par *POP3_SetPrefs* ou *MSG_SetPrefs*),
  Dans les deux cas, en l'absence de dossier défini, le dossier utilisé sera celui contenant le fichier de structure de la base (avec 4D monoposte) ou le dossier de l'application 4D Client (avec 4D Server).

Note de compatibilité (version 6.8.1) : Si la commande *MSG_SetPrefs* n'est pas utilisée, ce sont les paramètres *dossierMsg* et *dossierDocsJoints* de la commande *POP3_SetPrefs* qui seront pris en compte — si cette dernière a été préalablement exécutée. Si la commande *MSG_SetPrefs* est utilisée, les paramètres *dossierMsg* et *dossierDocsJoints* de la commande *POP3_SetPrefs* seront ignorés.

Si vous ne passez ni 0 ni 1 dans le paramètre *dossier*, par défaut la valeur 0 est utilisée.

Attention : Cette commande est à utiliser avec précaution, elle supprime **TOUT** fichier qui lui est transmis.

MSG_Extract

MSG_Extract (nomFichier ; décoder ; cheminDocsJoints ; listePiècesJointes) -> Résultat

Paramètre	Type		Description
nomFichier	Texte	➡	Nom de fichier
décoder	Entier	➡	0 = Pas de décodage, 1 = Décoder si possible
cheminDocsJoints	Texte	➡	Chemin du dossier (chemin par défaut dans le dossier DocsJoints)
listePiècesJointes	Tableau chaîne	⬅	Noms des fichiers joints (sans chemins d'accès)
Résultat	Entier	↩	Code d'erreur

Description

La commande *MSG_Extract* extrait tous les documents joints et les place dans le dossier des documents joints.

nomFichier désigne le nom ou le chemin d'accès complet du fichier duquel extraire les documents joints. Si vous passez un nom de fichier seul, le chemin d'accès par défaut sera celui du dossier défini par *POP3_SetPrefs* ou *MSG_SetPrefs* (voir Note de compatibilité). En l'absence de dossier spécifié, le chemin par défaut sera celui du dossier contenant le fichier de structure de la base de données (avec 4D monoposte) ou du dossier de 4D Client (avec 4D Server).

Le paramètre *décoder* spécifie si une tentative de décodage du ou des documents joints doit être effectuée. Si vous passez 0 (zéro), aucune tentative ne sera effectuée. Si vous passez 1, la commande décodera les fichiers, s'ils ont été encodés au moyen de l'une des méthodes suivantes : Binhex, AppleSingle, AppleDouble ou Base64.

Le paramètre *cheminDocsJoints* indique le chemin d'accès du dossier dans lequel enregistrer le document joint. Si vous passez une chaîne vide, le fichier est enregistré dans le dossier des documents joints spécifié par *POP3_SetPrefs* ou *MSG_SetPrefs* (voir Note de compatibilité). En l'absence de dossier spécifié, le fichier est enregistré dans le même dossier que celui du fichier de structure de la base de données.

Note de compatibilité (version 6.8.1) : Si la commande *MSG_SetPrefs* n'est pas utilisée, ce sont les paramètres *dossierMsg* et *dossierDocsJoints* de la commande *POP3_SetPrefs* qui seront pris en compte — si cette dernière a été préalablement exécutée. Si la commande *MSG_SetPrefs* est utilisée, les paramètres *dossierMsg* et *dossierDocsJoints* de la commande *POP3_SetPrefs* seront ignorés.

Le tableau alphanumérique/texte *listePiècesJointes* retourne les noms de tous les documents joints. Seul le nom du document est renvoyé dans l'élément du tableau, sans le chemin d'accès.

MSG_FindHeader

MSG_FindHeader (nomFichier ; libelléEnTête ; valeurEnTête) -> Résultat

Paramètre	Type		Description
nomFichier	Texte	⇒	Nom de fichier
libelléEnTête	Chaîne	⇒	Libellé de l'en-tête ("From:", "To:", "Subject:", etc.)
valeurEnTête	Texte	⇐	Valeur
Résultat	Entier	↩	Code d'erreur

Description

La commande *MSG_FindHeader* recherche le *libelléEnTête* dans la section d'en-tête de *nomFichier* et retourne la valeur du champ dans *valeurEnTête*. *nomFichier* contient le nom et/ou le chemin d'accès d'un message téléchargé localement par la commande *POP3_DownLoad* ou *IMAP_Download*.

nomFichier désigne le nom ou le chemin d'accès complet du fichier duquel extraire la valeur d'en-tête. Si vous passez un nom de fichier seul, le chemin d'accès par défaut sera celui du dossier défini par *POP3_SetPrefs* ou *MSG_SetPrefs* (voir Note de compatibilité). En l'absence de dossier spécifié, le chemin par défaut sera celui du dossier contenant le fichier de structure de la base de données (avec 4D monoposte) ou du dossier de 4D Client (avec 4D Server).

Note de compatibilité (version 6.8.1) : Si la commande *MSG_SetPrefs* n'est pas utilisée, le paramètre *dossierMsg* éventuellement défini par la commande *POP3_SetPrefs* est pris en compte. Si la commande *MSG_SetPrefs* est utilisée, le paramètre défini par la commande *POP3_SetPrefs* est ignoré.

Passez dans *libelléEnTête* une chaîne de caractères contenant le libellé d'en-tête à rechercher. *libelléEnTête* peut faire référence à tout en-tête défini, spécifié par l'utilisateur ou étendu, tel que "From:" (Emetteur), "To:" (Destinataire), "X-MonEnTête", etc.

Note : Les libellés des en-têtes sont toujours exprimés en anglais.

Le paramètre *valeurEnTête* reçoit la valeur affectée à la zone d'en-tête spécifiée. Ce paramètre étant susceptible de contenir des caractères étendus, vous pouvez automatiser la gestion de ceux-ci à l'aide de la commande *POP3_Charset* ou *MSG_Charset*.

Note de compatibilité (version 6.8.1) : Si la commande *MSG_Charset* n'est pas utilisée, le paramètre *jeuCorps* éventuellement défini par la commande *POP3_Charset* est pris en compte. Si la commande *MSG_Charset* est utilisée, le paramètre défini par la commande *POP3_Charset* est ignoré.

MSG_GetBody

MSG_GetBody (nomFichier ; décalage ; longueur ; texteCorps) -> Résultat

Paramètre	Type		Description
nomFichier	Texte	⇒	Nom de fichier
décalage	Entier long	⇒	Début du décalage dans le corps du message (0 = début du corps)
longueur	Entier long	⇒	Nombre de caractères
texteCorps	Texte	⇐	Texte du corps (supprime les retours à la ligne si Prefs ON)
Résultat	Entier long	⇐	Code d'erreur

Description

La commande *MSG_GetBody* renvoie uniquement le texte du corps du message désigné par *nomFichier*. Elle n'inclut pas le texte des pièces jointes et supprime tous les en-têtes MIME.

nomFichier désigne le nom ou le chemin d'accès complet du fichier duquel extraire le corps du message. Si vous passez un nom de fichier seul, le chemin d'accès par défaut sera celui du dossier défini par *POP3_SetPrefs* ou *MSG_SetPrefs* (voir Notes de compatibilité). En l'absence de dossier spécifié, le chemin par défaut sera celui du dossier contenant le fichier de structure de la base de données (avec 4D monoposte) ou du dossier de 4D Client (avec 4D Server).

Le paramètre *décalage* vous permet de définir, dans le corps, la position du caractère à partir duquel commencer la récupération.

Le paramètre *longueur* indique le nombre de caractères à renvoyer.

Le paramètre *texteCorps* retourne le texte du corps du message. Ce paramètre étant susceptible de contenir des caractères étendus, vous pouvez automatiser la gestion de ceux-ci à l'aide de la commande *POP3_Charset* ou *MSG_Charset* (voir Notes de compatibilité).
Par ailleurs, ce paramètre tient compte de la valeur du paramètre *retoursLigne* éventuellement défini par *POP3_SetPrefs* ou *MSG_SetPrefs* (voir Notes de compatibilité).

Notes de compatibilité (version 6.8.1) :

- Si la commande *MSG_SetPrefs* n'est pas utilisée, les paramètres *dossierMsg* et *retoursLigne* éventuellement définis par la commande *POP3_SetPrefs* sont pris en compte. Si la commande *MSG_SetPrefs* est utilisée, les paramètres définis par la commande *POP3_SetPrefs* sont ignorés.
- Si la commande *MSG_Charset* n'est pas utilisée, le paramètre *jeuCorps* éventuellement défini par la commande *POP3_Charset* est pris en compte. Si la commande *MSG_Charset* est utilisée, le paramètre *jeuCorps* défini par la commande *POP3_Charset* est ignoré.

MSG_GetHeaders

MSG_GetHeaders (nomFichier ; décalage ; longueur ; texteEnTête) -> Résultat

Paramètre	Type		Description
nomFichier	Texte	→	Nom de fichier
décalage	Entier long	→	Début du décalage dans la section d'en-têtes (0 = début de l'en-tête)
longueur	Entier long	→	Nombre de caractères
texteEnTête	Texte	←	Texte de la section d'en-têtes (supprime les retours à la ligne si Prefs ON)
Résultat	Entier	↩	Code d'erreur

Description

La commande *MSG_GetHeaders* renvoie sous forme de texte brut la section d'en-têtes du message désigné par *nomFichier*. L'en-tête d'un message POP3 comprend tout le texte situé à partir du début du message, jusqu'à la première occurrence de deux séquences consécutives de retour chariot/retour ligne (carriage return/line feed).

nomFichier désigne le nom ou le chemin d'accès complet du fichier duquel extraire l'en-tête. Si vous passez un nom de fichier seul, le chemin d'accès par défaut sera celui du dossier défini par *POP3_SetPrefs* ou *MSG_SetPrefs* (voir Note de compatibilité). En l'absence de dossier spécifié, le chemin par défaut sera celui du dossier contenant le fichier de structure de la base de données (avec 4D monoposte) ou du dossier de 4D Client (avec 4D Server).

Le paramètre *décalage* vous permet de définir, dans la section d'en-tête source, la position du caractère à partir duquel commencer la récupération.

Le paramètre *longueur* définit le nombre de caractères à renvoyer. La longueur de la section d'en-têtes peut être récupérée via *MSG_MessageSize*.

Le paramètre *texteEnTête* retourne le texte brut de l'en-tête. Ce paramètre tient compte de la valeur du paramètre *retoursLigne* éventuellement défini par *POP3_SetPrefs* ou *MSG_SetPrefs*.

Note de compatibilité (version 6.8.1) : Si la commande *MSG_SetPrefs* n'est pas utilisée, les paramètres *dossierMsg* et *retoursLigne* éventuellement définis par la commande *POP3_SetPrefs* sont pris en compte. Si la commande *MSG_SetPrefs* est utilisée, les paramètres définis par la commande *POP3_SetPrefs* sont ignorés.

MSG_GetMessage

MSG_GetMessage (nomFichier ; décalage ; longueur ; texteBrut) -> Résultat

Paramètre	Type		Description
nomFichier	Texte	➡	Nom de fichier
décalage	Entier long	➡	Début du décalage dans le fichier du message (0 = début du fichier)
longueur	Entier long	➡	Nombre de caractères
texteBrut	Texte	⬅	Texte brut (ignore les Prefs)
Résultat	Entier	↩	Code d'erreur

Description

La commande *MSG_GetMessage* renvoie le texte brut du message désigné par *nomFichier*, quelles que soient les pièces jointes. Elle ne supprime pas les en-têtes MIME.

nomFichier désigne le nom ou le chemin d'accès complet du fichier duquel extraire le corps du message. Si vous passez un nom de fichier seul, le chemin d'accès par défaut sera celui du dossier défini par *POP3_SetPrefs* ou *MSG_SetPrefs* (voir Note de compatibilité). En l'absence de dossier spécifié, le chemin par défaut sera celui du dossier contenant le fichier de structure de la base de données (avec 4D monoposte) ou du dossier de 4D Client (avec 4D Server).

Note de compatibilité (version 6.8.1) : Si la commande *MSG_SetPrefs* n'est pas utilisée, le paramètre *dossierMsg* éventuellement défini par la commande *POP3_SetPrefs* est pris en compte. Si la commande *MSG_SetPrefs* est utilisée, le paramètre défini par la commande *POP3_SetPrefs* est ignoré.

Le paramètre *décalage* vous permet de définir, dans le message source, la position du caractère à partir duquel commencer la récupération.

Le paramètre *longueur* indique le nombre de caractères à renvoyer.

Le paramètre *texteBrut* retourne l'intégralité du texte du message. Les paramètres des préférences pour la suppression des retours à la ligne (line feed) sont ignorés et les éventuels documents joints imbriqués dans le corps du message ne sont pas supprimés.

MSG_GetPrefs

MSG_GetPrefs (retoursLigne ; dossierMsg ; dossierDocsJoints) -> Résultat

Paramètre	Type		Description
retoursLigne	Entier	⇐	0 = Ne pas retirer les retours à la ligne, 1 = Retirer les retours à la ligne
dossierMsg	Texte	⇐	Chemin d'accès au dossier des messages ("" = aucune modification)
dossierDocsJoints	Texte	⇐	Chemin d'accès du dossier des documents joints ("" = aucune modification)
Résultat	Entier	⇒	Code d'erreur

Description

La commande *MSG_GetPrefs* permet de connaître les préférences courantes pour les commandes MSG.

Le paramètre *retoursLigne* retourne le paramétrage courant de l'option de suppression des retours à la ligne.

Le paramètre *dossierMsg* retourne le chemin d'accès local du dossier dans lequel sont enregistrés par défaut les messages récupérés.

Le paramètre *dossierDocsJoints* retourne le chemin d'accès local du dossier dans lequel sont enregistrés par défaut les documents joints extraits des messages.

MSG_HasAttach

MSG_HasAttach (nomFichier ; nbreDocsJoints) -> Résultat

Paramètre	Type		Description
nomFichier	Texte	➡	Nom de fichier
nbreDocsJoints	Entier	⬅	Nombre de documents joints
Résultat	Entier	↩	Code d'erreur

Description

La commande *MSG_HasAttach* retourne dans le paramètre *nbreDocsJoints* le nombre de documents joints au message désigné par *nomFichier*. Un document joint est une pièce jointe qui n'est pas en texte MIME. Si le message n'a pas de document joint, la commande retourne 0 dans *nbreDocsJoints*.

nomFichier désigne le nom ou le chemin d'accès complet du fichier dans lequel vérifier la présence de documents joints. Si vous passez un nom de fichier seul, le chemin d'accès par défaut sera celui du dossier défini par *POP3_SetPrefs* ou *MSG_SetPrefs* (voir Note de compatibilité). En l'absence de dossier spécifié, le chemin par défaut sera celui du dossier contenant le fichier de structure de la base de données (avec 4D monoposte) ou du dossier de 4D Client (avec 4D Server).

Note de compatibilité (version 6.8.1) : Si la commande *MSG_SetPrefs* n'est pas utilisée, le paramètre *dossierMsg* éventuellement défini par la commande *POP3_SetPrefs* est pris en compte. Si la commande *MSG_SetPrefs* est utilisée, le paramètre défini par la commande *POP3_SetPrefs* est ignoré.

Le paramètre *nbreDocsJoints* retourne le nombre de documents joints à *nomFichier*.

MSG_MessageSize

MSG_MessageSize (nomFichier ; tailleEnTête ; tailleCorps ; tailleMsg) -> Résultat

Paramètre	Type		Description
nomFichier	Texte	→	Nom de fichier
tailleEnTête	Entier long	←	Taille de la section d'en-tête (soustrait les retours à la ligne si Prefs ON)
tailleCorps	Entier long	←	Taille du corps (soustrait les retours à la ligne si Prefs ON)
tailleMsg	Entier long	←	Taille du message entier ou du fichier (ignore les Prefs)
Résultat	Entier	↩	Code d'erreur

Description

La commande *MSG_MessageSize* renvoie des informations sur la taille des différentes parties du message désigné par *nomFichier*. *nomFichier* contient le nom et/ou le chemin d'accès d'un message téléchargé localement par la commande *POP3_DownLoad*.

nomFichier désigne le nom ou le chemin d'accès complet du fichier duquel extraire les informations du message. Si vous passez un nom de fichier seul, le chemin d'accès par défaut sera celui du dossier défini par *POP3_SetPrefs* ou *MSG_SetPrefs* (voir Note de compatibilité). En l'absence de dossier spécifié, le chemin par défaut sera celui du dossier contenant le fichier de structure de la base de données (avec 4D monoposte) ou du dossier de 4D Client (avec 4D Server).

Le paramètre *tailleEnTête* retourne la taille de la section d'en-tête.
Le paramètre *tailleCorps* retourne la taille du corps du texte.
Ces deux paramètres tiennent compte de la valeur du paramètre *retoursLigne* éventuellement défini par *POP3_SetPrefs* ou *MSG_SetPrefs*.

Note de compatibilité (version 6.8.1) : Si la commande *MSG_SetPrefs* n'est pas utilisée, les paramètres *dossierMsg* et *retoursLigne* éventuellement définis par la commande *POP3_SetPrefs* sont pris en compte. Si la commande *MSG_SetPrefs* est utilisée, les paramètres définis par la commande *POP3_SetPrefs* sont ignorés.

Le paramètre *tailleMsg* retourne la taille globale du message.

MSG_SetPrefs

MSG_SetPrefs (retoursLigne ; dossierMsg ; dossierDocsJoints) -> Résultat

Paramètre	Type		Description
retoursLigne	Entier	➡	0 = Ne pas retirer les retours à la ligne, 1 = Retirer les retours à la ligne, -1 = Aucune modification
dossierMsg	Texte	➡	Chemin d'accès au dossier des messages ("" = aucune modification)
dossierDocsJoints	Texte	➡	Chemin d'accès du dossier des documents joints ("" = aucune modification)
Résultat	Entier	↩	Code d'erreur

Description

La commande *MSG_SetPrefs* définit des préférences générales pour toutes les commandes MSG ultérieures.

Le paramètre *retoursLigne* vous permet de préciser comment traiter les caractères de retour à la ligne dans les messages téléchargés. La plupart des messages Internet associent un caractère Retour chariot (Carriage return) et un caractère Retour à la ligne (Line feed) pour indiquer la fin d'une ligne, à la différence des applications Macintosh qui requièrent un simple Retour chariot. Dans ce cas, cette option vous permet de supprimer les caractères Retour à la ligne superflus du texte des messages.

- Si vous passez 0 (zéro) les messages récupérés seront conservés dans leur format initial.
- Si vous passez 1, les caractères Retour à la ligne seront supprimés des messages récupérés.
- Si vous passez -1, l'option reste telle qu'elle était précédemment définie.

Par défaut, cette option a pour valeur 1, les retours à la ligne rencontrés dans les messages sont automatiquement supprimés.

Le paramètre *dossierMsg* indique le chemin d'accès local du dossier dans lequel les messages récupérés doivent être enregistrés par défaut.

Note de compatibilité (version 6.8.1) : Si la commande *MSG_SetPrefs* n'est pas utilisée, ce sont les paramètres *retoursLigne* et *dossierMsg* de la commande *POP3_SetPrefs* qui seront pris en compte — si cette dernière a été préalablement exécutée. Si la commande *MSG_SetPrefs* est utilisée, les paramètres *retoursLigne* et *dossierMsg* de la commande *POP3_SetPrefs* seront ignorés.

Le paramètre *dossierDocsJoints* indique le chemin d'accès local du dossier dans lequel les fichiers joints doivent être enregistrés lorsque la commande *MSG_Extract* extrait les documents joints du corps du message.

Note de compatibilité (version 6.8.1) : Ce paramètre est également présent dans la commande *POP3_SetPrefs*, par conséquent vous pouvez le fixer à l'aide de l'une de ces deux commandes. L'usage de la commande *MSG_SetPrefs* est toutefois fortement recommandé. Le paramètre de la commande *POP3_SetPrefs*, conservé pour des raisons de compatibilité, ne sera plus utilisé dans les prochaines versions du plug-in (ce paramètre est désormais optionnel). Cette recommandation s'applique également à la commande *POP3_GetPrefs*.

IC File Transfer

- Transfert de fichiers, Présentation
- FTP_Append
- FTP_ChangeDir
- FTP_Delete
- FTP_GetDirList
- FTP_GetFileInfo
- FTP_GetPassive
- FTP_GetType
- FTP_Login
- FTP_Logout
- FTP_MacBinary
- FTP_MakeDir
- FTP_PrintDir
- FTP_Receive
- FTP_RemoveDir
- FTP_Rename
- FTP_Send
- FTP_SetPassive
- FTP_SetType
- FTP_System
- FTP_VerifyID
- *FTP_Progress*

Transfert de fichiers, Présentation

Le protocole FTP (File Transfer Protocol) est le principal moyen de transférer documents et applications d'un ordinateur à un autre. Les "sites" FTP sont des ordinateurs dispersés dans le monde qui exécutent un logiciel FTP serveur. Le protocole FTP permet d'échanger des fichiers entre des systèmes disparates. Des applications clientes sur différentes plates-formes peuvent se connecter à un serveur FTP pour télécharger ou envoyer des fichiers texte ou binaires.

Les commandes FTP de 4D fournissent aux développeurs des outils leur permettant de créer des clients FTP à l'intérieur de leurs bases de données 4D.

Notes :

- Lors de la spécification des chemins d'accès dans les commandes FTP, vous devez toujours définir les emplacements de fichiers sur les sites FTP comme étant des répertoires de type Unix, même si le serveur FTP est un Macintosh. Quelle que soit la plate-forme utilisée, le logiciel du serveur FTP convertit en interne ce chemin d'accès au format requis pour envoyer ses documents aux clients connectés.
- Pour une plus grande souplesse, les commandes Internet de 4D permettent de passer directement une référence de connexion FTP aux commandes TCP de bas niveau et inversement. Pour plus d'informations, reportez-vous à la section **Routines de bas niveau, Présentation**.
- Dans 4D Internet Commands v16 R2 et suivantes, la taille des noms de fichiers manipulés par les routines est limitée à 1024 caractères, quel que soit l'OS(*).
  (*) Dans les versions macOS 32 bits, 4D Internet Commands doit utiliser des APIs de gestion de fichiers déclarées obsolètes et non maintenues par Apple, Inc. Dans cet environnement, suivant la version de macOS, les limitations relatives à la longueur des noms de fichiers (ainsi qu'aux caractères qu'ils acceptent) peuvent être plus restrictives. Nous recommandons fortement de mettre à niveau 4D Internet Commands en version 64 bits dès que possible.

FTP_Append

FTP_Append (ftp_ID ; cheminLocal ; cheminServeur ; progression) -> Résultat

Paramètre	Type		Description
ftp_ID	Entier long	⇒	Référence d'une connexion FTP
cheminLocal	Texte	⇒	Chemin d'accès local du document à envoyer
cheminServeur	Texte	⇒	Chemin d'accès du document sur le serveur FTP
progression	Entier	⇒	*** Paramètre obsolète (ignoré) ***
Résultat	Entier	⇐	Code d'erreur

Description

La commande *FTP_Append* effectue la même action que *FTP_Send*, à la différence près qu'elle ajoute les données envoyées à la fin du fichier existant identifié par *cheminServeur*. La principale fonction de cette commande est d'ajouter des données à la fin de fichiers texte préexistants.

FTP_ChangeDir

FTP_ChangeDir (ftp_ID ; cheminServeur) -> Résultat

Paramètre	Type		Description
ftp_ID	Entier long	➡	Référence d'une connexion FTP
cheminServeur	Texte	➡	Chemin d'accès à un répertoire Unix sur le serveur FTP
Résultat	Entier	⮌	Code d'erreur

Description

La commande *FTP_ChangeDir* permet de désigner le répertoire de travail courant (Current Working Directory ou CWD) sur le serveur FTP.

Note : Il est également possible de modifier le répertoire de travail courant à l'aide des commandes *FTP_GetDirList* et *FTP_GetFileInfo*. Toutefois, la commande *FTP_ChangeDir* est plus rapide et nécessite moins de paramètres.

ftp_ID est l'identifiant de la session FTP établie avec *FTP_Login*.

Le paramètre *cheminServeur* contient un chemin d'accès au format Unix référençant un répertoire FTP existant et accessible. Si le répertoire FTP spécifié est invalide (inexistant ou droits d'accès insuffisants), *FTP_ChangeDir* retourne une erreur et ne modifie pas le répertoire de travail courant.

Exemple

L'exemple suivant désigne la racine du serveur FTP comme répertoire de travail courant:

```
$err:=FTP_ChangeDir($ftp_ID;"/")
```

FTP_Delete

FTP_Delete (ftp_ID ; cheminServeur) -> Résultat

Paramètre	Type		Description
ftp_ID	Entier long	⇒	Référence d'une connexion FTP
cheminServeur	Texte	⇒	Chemin d'accès du document sur le serveur FTP
Résultat	Entier	⇐	Code d'erreur

Description

La commande *FTP_Delete* supprime le fichier désigné par *cheminServeur* du serveur FTP distant. Une erreur est renvoyée si vous n'avez pas les droits d'accès requis pour effectuer cette opération.

ftp_ID est l'identifiant de la session FTP établie avec *FTP_Login*.

cheminServeur désigne le chemin d'accès au format Unix du document à supprimer. Vous pouvez passer dans ce paramètre un chemin d'accès complet ou un simple nom de fichier. Si vous utilisez la forme abrégée, le fichier spécifié doit se trouver dans le répertoire de travail courant (CWD).

Note : La commande *FTP_ChangeDir* permet de modifier le répertoire de travail courant (CWD). Vous pouvez également connaître à tout moment le répertoire de travail courant à l'aide de la commande *FTP_PrintDir*.

FTP_GetDirList

FTP_GetDirList (ftp_ID ; cheminServeur ; tabNoms ; tailles ; types ; datesModif {; heuresModif}) -> Résultat

Paramètre	Type		Description
ftp_ID	Entier long	⇒	Référence d'une connexion FTP
cheminServeur	Texte	⇒	Chemin d'accès à un répertoire Unix sur le serveur FTP
		⇐	Répertoire de travail courant (CWD)
tabNoms	Tableau chaîne	⇐	Liste de noms
tailles	Tableau entier long	⇐	Liste de tailles
types	Tableau entier	⇐	Liste de types 0 = fichier normal, 1 = répertoire, 2 = fichier spécial de type bloc, 3 = fichier spécial de type caractère, 4 = lien symbolique, 5 = fichier spécial FIFO, 6 = porte d'accès de la famille d'adresses AF_UNIX
datesModif	Tableau date	⇐	Liste des dates de modification
heuresModif	Tableau entier long	⇐	Liste des heures de modification
Résultat	Entier	↩	Code d'erreur

Description

La commande *FTP_GetDirList* retourne la liste des objets présents dans le répertoire *cheminServeur* de la session FTP identifiée par *ftp_ID*. Le nom, la taille, le type, la date et, facultativement, l'heure de modification des éléments du répertoire *cheminServeur* sont renvoyés dans des tableaux. Une connexion au site FTP doit avoir déjà été ouverte par *FTP_Login* et être toujours valide (*FTP_VerifyID*). La commande *FTP_GetDirList* remplace le répertoire de travail courant (ou CWD, Current Working Directory) par celui défini dans le paramètre *cheminServeur*.

ftp_ID est l'identifiant de la session FTP établie avec *FTP_Login*.

Le paramètre *cheminServeur* contient un chemin d'accès au format Unix référençant un répertoire FTP. Il est fortement recommandé de passer une variable ou un champ 4D dans ce paramètre car le répertoire de travail courant résultant lui sera retourné après l'exécution de la commande. Généralement, la valeur renvoyée sera identique à celle qui a été passée. Toutefois, dans certains cas (par exemple des restrictions d'accès), le changement de répertoire a échoué. Le paramètre *cheminServeur* contient alors le chemin d'accès du répertoire courant du serveur pour la session.
Si vous passez une chaîne vide dans ce paramètre, les tableaux sont remplis avec la liste des fichiers du répertoire courant et le chemin d'accès du répertoire courant du serveur (CWD) est retourné dans le paramètre *cheminServeur*.

noms est un tableau de type alphanumérique ou texte recevant le nom de chaque objet présent dans le répertoire *cheminServeur* spécifié.

tailles est un tableau de type entier long recevant la taille des objets du répertoire *cheminServeur*.

types est un tableau de type entier recevant les valeurs de type de chaque objet du répertoire *cheminServeur*. Voici les valeurs possibles et les types correspondants :

Type	Fichier
0	fichier ordinaire
1	répertoire
2	fichier spécial de type bloc
3	fichier spécial de type caractère
4	lien symbolique (alias vers des fichiers ou des dossiers)
5	fichier spécial FIFO
6	porte d'accès de la famille AF_UNIX

Note : Dans le cas d'un lien symbolique (*type*=4), le serveur FTP retourne un chemin d'accès particulier (Nom d'alias + symbole + chemin d'accès au fichier ou dossier source). Si vous tentez d'utiliser ce chemin pour accéder au fichier ou dossier source, une erreur sera retournée. Vous devez extraire le chemin d'accès du fichier ou du dossier à partir de la chaîne retournée par *FTP_GetDirList*. Ce chemin d'accès débute immédiatement après le caractère symbole. Sinon, les commandes telles que *FTP_GetFileInfo* retourneront l'erreur -10085 puisque le fichier ou le dossier ne sera pas trouvé.

datesModif est un tableau de type date recevant la date de dernière modification de chaque objet du répertoire *cheminServeur*.

heuresModif est un tableau de type entier long recevant l'heure de dernière modification de chaque objet du répertoire *cheminServeur*.

Rappel : Dans 4D, le type de tableau entier long est utilisé pour manipuler les données de type heure. Chaque élément du tableau représente un nombre de secondes. Utilisez la commande **Chaine heure** pour convertir ces valeurs au format **HH:MM:SS**.

FTP_GetFileInfo

FTP_GetFileInfo (ftp_ID ; cheminServeur ; taille ; dateModif ; heureModif) -> Résultat

Paramètre	Type		Description
ftp_ID	Entier long	⇒	Référence d'une connexion FTP
cheminServeur	Texte	⇒	Chemin d'accès au document sur le serveur FTP
taille	Entier long	⇐	Taille du document
dateModif	Date	⇐	Date de modification
heureModif	Heure	⇐	Heure de modification
Résultat	Entier	↩	Code d'erreur

Description

La commande *FTP_GetFileInfo* retourne des informations concernant la dernière modification du fichier désigné par *cheminServeur*.

ftp_ID est l'identifiant de la session FTP établie avec *FTP_Login*.

cheminServeur contient le chemin d'accès au document sur lequel vous souhaitez obtenir des informations.

Note : La commande *FTP_GetFileInfo* peut modifier le répertoire de travail courant (CWD) si *cheminServeur* est un chemin d'accès complet qui indique un répertoire différent du répertoire de travail courant. Dans ce cas, le répertoire défini par le paramètre *cheminServeur* devient le répertoire de travail courant.

Le paramètre *taille* retourne la taille du fichier identifié par *cheminServeur*.

Les paramètres *dateModif* et *heureModif* retournent la date et l'heure de la dernière modification du fichier.

FTP_GetPassive

FTP_GetPassive (ftp_ID ; modePassif) -> Résultat

Paramètre	Type		Description
ftp_ID	Entier long	→	Référence d'une connexion FTP
modePassif	Entier	←	Mode d'échange en cours 0 = mode actif, 1 = mode passif
Résultat	Entier	↩	Code d'erreur

Description

La commande *FTP_GetPassive* permet de connaître le mode de transfert en vigueur sur le canal de transfert des données. Pour plus d'informations sur ce paramétrage, reportez-vous à la description de la commande *FTP_SetPassive*.

ftp_ID est l'identifiant de la session FTP établie avec *FTP_Login*.

Le paramètre *modePassif* retourne le mode de transfert courant sur le canal de transfert des données :

- si *modePassif* vaut 0, le serveur FTP fonctionne en mode actif.
- si *modePassif* vaut 1, le serveur FTP fonctionne en mode passif (mode par défaut).

FTP_GetType

FTP_GetType (ftp_ID ; ftp_Mode) -> Résultat

Paramètre	Type		Description
ftp_ID	Entier long	⇒	Référence d'une connexion FTP
ftp_Mode	Chaîne	⇐	"A" = Ascii ; "I" = Image ; "L 8" = 8 bits Logique
Résultat	Entier	⇒	Code d'erreur

Description

La commande *FTP_GetType* renvoie des informations sur le mode de transfert FTP courant. Le mode de transfert peut être fixé au moyen de la commande *FTP_SetType*.

ftp_ID est l'identifiant de la session FTP établie avec *FTP_Login*.

Le paramètre *ftp_Mode* retourne un code indiquant le mode de transfert FTP courant.

FTP_Login

FTP_Login (nomServeur ; nomUtilisateur ; motDePasse ; ftp_ID ; texteAccueil) -> Résultat

Paramètre	Type		Description
nomServeur	Chaîne	➡	Nom ou adresse IP du serveur FTP
nomUtilisateur	Chaîne	➡	Nom de l'utilisateur
motDePasse	Chaîne	➡	Mot de passe
ftp_ID	Entier long	⬅	Référence de cette nouvelle session FTP
texteAccueil	Texte	⬅	Texte d'accueil FTP
Résultat	Entier	↩	Code d'erreur

Description

La commande *FTP_Login* établit une connexion avec le serveur FTP *nomServeur* et se connecte au système au moyen des *nomUtilisateur* et *motDePasse* fournis.

Le paramètre *nomServeur* contient le nom ou l'adresse IP du serveur distant.

nomUtilisateur contient le nom du compte utilisateur reconnu par le serveur FTP distant. De nombreux serveurs FTP acceptent l'accès d'invités au moyen du nom d'utilisateur "anonymous". Si vous vous connectez de façon anonyme, il est de règle de fournir votre adresse e-mail comme mot de passe.

Le paramètre *motDePasse* contient le mot de passe de *nomUtilisateur* sur le serveur FTP.

Le paramètre *ftp_ID* retourne un entier long identifiant la session ouverte. Cette valeur sera utilisée par les commandes FTP suivantes.

Le paramètre optionnel *texteAccueil* récupère le texte envoyé par le serveur FTP lorsque l'utilisateur se connecte. En effet, de nombreux sites FTP envoient un message d'accueil lors de la connexion.

Exemple

```
$OK:=False
Case of
   :(FTP_Login("ftp.4d.com";"anonymous";"dbody@aol.com";vFTP_ID;vFTP_TxtAccueil)#0)
   :(FTP_Send(vFTP_ID;"Mon disque dur:Documents ƒ:Rapport des ventes de juillet";"/pub/rapports";1)#0)
   :(FTP_Logout(vFTP_ID)#0)
   Else
      $OK:=True  `Toutes les commandes ont été exécutées sans erreur
End case
```

Note : Pour plus d'informations sur cet emploi particulier de la structure **Au cas ou**, reportez-vous à l'**Annexe A, Conseils de programmation**.

FTP_Logout

FTP_Logout (ftp_ID) -> Résultat

Paramètre	Type		Description
ftp_ID	Entier long	➡	Référence d'une connexion FTP
		⬅	0 = Session correctement fermée
Résultat	Entier	↩	Code d'erreur

Description

La commande *FTP_Logout* déconnecte l'utilisateur du serveur FTP et libère la mémoire utilisée pour la session FTP désignée par *ftp_ID*. Après son exécution, la commande retourne la valeur 0 (zéro) dans le paramètre *ftp_ID* si la fermeture de la session a réussi.

ftp_ID est l'identifiant de la session FTP établie avec *FTP_Login*.

Exemple

```
If(FTP_Login("ftp.4d.com";"anonymous";"dbody@aol.com";vFTP_ID;vTxt_Accueil)=1)
    $erreur:=FTP_Send(vFTP_ID;"Mon disque dur:Documents:Rapports des ventes";"/pub/rapports";1)
    $erreur:=FTP_Logout(vFTP_ID)
End if
```

FTP_MacBinary

FTP_MacBinary (ftp_ID ; modeMacBinary) -> Résultat

Paramètre	Type		Description
ftp_ID	Entier long	➡	Référence d'une connexion FTP
modeMacBinary	Entier	➡	-1 = Obtenir le paramètre courant, 1 = Activer, 0 = Désactiver
		⬅	Paramètre courant (si -1 passé)
Résultat	Entier	↩	Code d'erreur

Description

La commande *FTP_MacBinary* active/désactive le mode MacBinary lors des transferts FTP utilisant *FTP_Send* ou *FTP_Receive* dans la session FTP courante identifiée par *ftp_ID*.

Le protocole MacBinary est souvent utilisé par les clients et serveurs FTP Macintosh pour faciliter le transfert de données ou de fichiers binaires contenant à la fois des data forks (parties de "données") et des resource forks (parties de "ressources").

Note à l'attention des utilisateurs Windows : Il est possible d'utiliser le protocole MacBinary pour des transferts FTP dans un environnement Windows. Toutefois, il est généralement inutile de décoder un fichier MacBinary sur un PC. Les ordinateurs à base de processeur Intel ne peuvent pas stocker des fichiers contenant une data forks et une resource fork (format Mac OS). Comme ce format est étranger à la plate-forme PC, les fichiers Mac OS contenant une resource fork risquent d'être endommagés s'ils sont enregistrés dans un format non encodé.

ftp_ID est l'identifiant de la session FTP établie avec *FTP_Login*.

Le paramètre *modeMacBinary* indique s'il faut activer ou non le mode de transfert MacBinary. Cette valeur doit être fournie en tant que variable pour que la commande puisse renvoyer l'état des transferts MacBinary après une tentative de modification.

- 1 active le mode de transfert MacBinary.
- 0 (zéro) désactive le mode de transfert MacBinary.
- -1 retourne dans ce paramètre *modeMacBinary* le paramétrage courant du mode de transfert MacBinary (1 ou 0).

Attention : Tous les serveurs FTP ne gèrent pas le protocole MacBinary. S'il n'est pas géré, l'erreur 10053 est générée à chaque appel de la commande *FTP_MacBinary*, quelle que soit la valeur du paramètre *modeMacBinary*. Les comportements précédemment décrits ne s'appliquent alors pas.

Exemple

Cet exemple active le protocole MacBinary avant la réception d'un fichier FTP. Si le fichier a été correctement reçu avec MacBinary activé, il est alors décodé dans son format original et le document MacBinary reçu est supprimé.

```
vUseMacBin:=-1
$erreur:=FTP_MacBinary(vFTP_ID;vUseMacBin)
If($erreur=10053)
   MacBinaryEstAccepte:=False  `Le serveur ftp ne gère pas le protocole MacBinary
Else
   MacBinaryEstAccepte:=True
End if


vFichierLocal:=""
If(MacBinaryEstAccepte)
   vUseMacBin:=1
   $erreur:=FTP_MacBinary(vFTP_ID;vUseMacBin)  `Activation de MacBinary pour le téléchargement
End if
$erreur:=FTP_Receive(vFTP_ID;"MonApplication";vFichierLocal;cbShowTherm)
If($erreur=0)&(vUseMacBin=1)  `Si la réception est OK et le fichier est au format MacBinary
```

```
    vCheminDecode:=""
    If(IT_Decode(vFichierLocal;vCheminDecode;8)=0)  `Décodage MacBinary
        DELETE DOCUMENT(vFichierLocal)  `Si le décodage a réussi, alors supprimer le fichier source
    End if
End if
```

FTP_MakeDir

FTP_MakeDir (ftp_ID ; cheminServeur) -> Résultat

Paramètre	Type		Description
ftp_ID	Entier long	➡	Référence d'une connexion FTP
cheminServeur	Texte	➡	Chemin d'accès à un répertoire Unix sur le serveur FTP
Résultat	Entier	⮌	Code d'erreur

Description

La commande *FTP_MakeDir* crée un nouveau répertoire défini par le paramètre *cheminServeur*. Une erreur est retournée si vous n'avez pas les droits d'accès requis pour effectuer cette opération.

ftp_ID est l'identifiant de la session FTP établie avec *FTP_Login*.

cheminServeur désigne le chemin d'accès au format Unix du répertoire FTP à créer. Ce paramètre peut contenir un chemin d'accès complet ou un simple nom de dossier. Si la forme abrégée est utilisée, le répertoire est alors créé dans le répertoire de travail courant (CWD). Le nom du répertoire *cheminServeur* ne doit **pas** comporter d'espace vide.

Note : La commande *FTP_ChangeDir* permet de modifier le répertoire de travail courant (CWD). Vous pouvez également connaître à tout moment le répertoire de travail courant à l'aide de la commande *FTP_PrintDir*.

FTP_PrintDir

FTP_PrintDir (ftp_ID ; cheminServeur) -> Résultat

Paramètre	Type		Description
ftp_ID	Entier long	→	Référence d'une connexion FTP
cheminServeur	Texte	←	Chemin d'accès Unix à un répertoire du serveur FTP
Résultat	Entier	↩	Code d'erreur

Description

La commande *FTP_PrintDir* retourne le répertoire de travail courant (Current Working Directory, ou CWD) sur le serveur FTP.

Note : Cette information peut également être obtenue à l'aide de l'instruction **FTP_GetDirList (ftpID;"";...)** en passant une chaîne vide dans le paramètre *cheminServeur*. Toutefois, la commande *FTP_PrintDir* est plus rapide et nécessite moins de paramètres.

ftp_ID est l'identifiant de la session FTP établie avec *FTP_Login*.

Le paramètre *cheminServeur* retourne le chemin du répertoire de travail courant (CWD).

Exemple

L'emplacement du CWD est retourné dans la variable $Cwd:

```
$err:=FTP_PrintDir($myftpid;$Cwd)
```

FTP_Receive

FTP_Receive (ftp_ID ; cheminServeur ; cheminLocal ; progression) -> Résultat

Paramètre	Type		Description
ftp_ID	Entier long	➡	Référence d'une connexion FTP
cheminServeur	Texte	➡	Chemin d'accès sur le serveur FTP du document à recevoir
cheminLocal	Texte	➡	Chemin d'accès local de destination du document
		⬅	Chemin d'accès du document résultant (si "" passé)
progression	Entier	➡	*** Paramètre obsolète (ignoré) ***
Résultat	Entier	↩	Code d'erreur

Description

La commande *FTP_Receive* reçoit par FTP un fichier dont le chemin d'accès sur le serveur FTP est défini par *cheminServeur*. *FTP_Receive* retourne l'erreur -48 si le fichier existe déjà dans le répertoire de destination.

ftp_ID est l'identifiant de la session FTP établie avec *FTP_Login*.

Le paramètre *cheminServeur* spécifie le chemin d'accès Unix complet du document à recevoir. Si *cheminServeur* ne contient pas un chemin d'accès complet à un document, la commande retourne une erreur. Comme pour tous les chemins d'accès à des documents Unix, le chemin doit utiliser des barres obliques ("/") comme séparateurs. Pour plus d'informations sur ce point, reportez-vous à la section **Glossaire et terminologie**.

Le paramètre *cheminLocal* spécifie le chemin d'accès du document à créer localement.

- Si vous passez une chaîne vide dans ce paramètre, la boîte de dialogue standard d'enregistrement de document s'affichera — dans ce cas, le nom et le chemin d'accès du fichier sélectionné par l'utilisateur seront retournés dans la variable *cheminLocal*.
- Si vous passez uniquement un nom de fichier, le fichier sera enregistré dans le même dossier que le fichier de structure de la base de données (avec 4D monoposte) ou dans le dossier de 4D Client (avec 4D Server).

Comme pour tous les chemins d'accès à des documents locaux, les répertoires doivent être séparés par le délimiteur correspondant à la plate-forme utilisée. Pour plus d'informations, reportez-vous à la section **Glossaire et terminologie**.

Compatibilité : Le paramètre *progression* est obsolète et est ignoré s'il est passé.

Exemple

```
vUseMacBin:=-1
$erreur:=FTP_MacBinary(vFTP_ID;vUseMacBin)
If($erreur=10053)
   MacBinaryEstAccepte:=False  `Le serveur ftp ne gère pas le protocole MacBinary
Else
   MacBinaryEstAccepte:=True
End if

vFichierLocal:=""
If(MacBinaryEstAccepte)
   vUseMacBin:=1
   $erreur:=FTP_MacBinary(vFTP_ID;vUseMacBin)  `Active MacBinary pour le téléchargement
   $erreur:=FTP_Receive(vFTP_ID;"CGMiniViewer.hqx";vFichierLocal)
   If($erreur=0) & (vUseMacBin=1)
      vCheminDecode:=""
      If(IT_Decode(vFichierLocal;vCheminDecode;8)=0)  `Décoder MacBinary
         DELETE DOCUMENT(vFichierLocal)  `Si le décodage a réussi, supprimer le fichier source
      End if
   End if
End if
```

FTP_RemoveDir

FTP_RemoveDir (ftp_ID ; cheminServeur) -> Résultat

Paramètre	Type		Description
ftp_ID	Entier long	➡	Référence d'une connexion FTP
cheminServeur	Texte	➡	Chemin d'accès à un répertoire Unix sur le serveur FTP
Résultat	Entier	⮌	Code d'erreur

Description

La commande *FTP_RemoveDir* supprime le répertoire désigné par le paramètre *cheminServeur*. Une erreur est retournée si vous n'avez pas les droits d'accès requis pour effectuer cette opération. En outre, si vous tentez de supprimer un répertoire contenant des éléments, une erreur de sécurité sera également retournée.

ftp_ID est l'identifiant de la session FTP établie avec *FTP_Login*.

cheminServeur désigne le chemin d'accès au format Unix d'un répertoire FTP à supprimer. Ce paramètre peut contenir un chemin d'accès complet ou un simple nom de dossier. Si la forme abrégée est utilisée, le répertoire spécifié doit se trouver dans le répertoire de travail courant (CWD).

Note : La commande *FTP_ChangeDir* permet de modifier le répertoire de travail courant (CWD). Vous pouvez également connaître à tout moment le répertoire de travail courant à l'aide de la commande *FTP_PrintDir*.

FTP_Rename

FTP_Rename (ftp_ID ; cheminServeur ; nouvNom) -> Résultat

Paramètre	Type		Description
ftp_ID	Entier long	➡	Référence d'une connexion FTP
cheminServeur	Texte	➡	Chemin d'accès du document sur le serveur FTP
nouvNom	Texte	➡	Nouveau nom du document
Résultat	Entier	⮌	Code d'erreur

Description

La commande *FTP_Rename* renomme le fichier désigné par *cheminServeur* sur le serveur FTP distant. Une erreur est générée si vous n'avez pas les droits d'accès requis pour effectuer cette opération.

ftp_ID est l'identifiant de la session FTP établie avec *FTP_Login*.

cheminServeur désigne le chemin d'accès au format Unix du document à renommer. Vous pouvez passer dans ce paramètre un chemin d'accès complet ou un simple nom de fichier. Si vous utilisez la forme abrégée, le fichier spécifié doit se trouver dans le répertoire de travail courant (CWD).

Note : La commande *FTP_ChangeDir* permet de modifier le répertoire de travail courant (CWD). Vous pouvez également connaître à tout moment le répertoire de travail courant à l'aide de la commande *FTP_PrintDir*.

Le paramètre *nouvNom* contient le nom avec lequel vous souhaitez renommer le document distant.

FTP_Send

FTP_Send (ftp_ID ; cheminLocal ; cheminServeur ; progression) -> Résultat

Paramètre	Type		Description
ftp_ID	Entier long	➡	Référence d'une connexion FTP
cheminLocal	Texte	➡	Chemin d'accès local du document à envoyer
cheminServeur	Texte	➡	Chemin d'accès du document sur le serveur FTP
progression	Entier	➡	*** Paramètre obsolète (ignoré) ***
Résultat	Entier	⮌	Code d'erreur

Description

La commande *FTP_Send* envoie le document désigné par *cheminLocal* à l'emplacement désigné par *cheminServeur*. Si une erreur de statut de fichier FTP se produit, *FTP_Send* la retourne immédiatement.

ftp_ID est l'identifiant de la session FTP établie avec *FTP_Login*.

cheminLocal désigne le chemin d'accès local du document à envoyer. Si vous passez une chaîne vide, la boîte de dialogue standard d'ouverture de fichiers apparaît. Si vous passez un nom de fichier simple (sans chemin d'accès), la commande recherche ce fichier dans le dossier contenant le fichier de structure de la base de données (avec 4D monoposte) ou dans le dossier de 4D Client (avec 4D Server). Comme pour tous les chemins d'accès à des documents locaux, les noms des répertoires doivent être séparés par le délimiteur correspondant à la plate-forme. Pour plus d'informations sur ce point, reportez-vous à la section **Glossaire et terminologie**.

Note : La longueur des noms des documents manipulés par les commandes FTP est limitée. Pour plus d'informations, reportez-vous à la section **Transfert de fichiers, Présentation**.

cheminServeur désigne le chemin d'accès complet au format Unix du document à créer, nom du fichier compris. Le *cheminServeur* indique le nouveau nom du fichier une fois reçu par le serveur FTP. Si *cheminLocal* est une chaîne vide permettant à l'utilisateur de sélectionner un fichier sur disque, *cheminServeur* peut alors aussi être une chaîne vide, auquel cas le nom du fichier sélectionné sera utilisé.

Vous pouvez passer dans *cheminServeur* un chemin d'accès Unix complet ou simplement un nom de fichier :

- Si vous passez un chemin d'accès complet, le fichier sera placé dans le répertoire défini, qui deviendra alors le répertoire de travail courant (CWD).
- Si vous passez uniquement un nom de fichier, ou si des chaînes vides sont passées dans les paramètres *cheminLocal* et *cheminServeur*, le fichier sera alors envoyé dans le répertoire de travail courant (CWD).

Si le paramètre *cheminServeur* ne peut pas être correctement interprété, ou si l'utilisateur n'a pas les droits d'accès requis pour envoyer un fichier dans le répertoire distant, la commande retourne une erreur. Comme pour tous les chemins d'accès des documents Unix, le chemin doit être défini à l'aide de barres obliques ("/"). Pour plus d'informations sur ce point, reportez-vous à la section **Glossaire et terminologie**.

Note : Des limitations spécifiques relatives à la longueur ou aux caractères utilisables dans les noms de fichiers peuvent être imposées par le serveur FTP.

Compatibilité : Le paramètre *progression* est obsolète et est ignoré s'il est passé.

Exemple 1

```
$OK:=False
Case of
    :(FTP_Login("ftp.4d.com";"anonymous";"dbody@aol.com";vFTP_ID;vTxt_Accueil)#0)
    :(FTP_Send(vFTP_ID;"Mon disque dur:Documents:Rapport des ventes de
juillet";"/pub/rapports/ventes_juillet";1)#0)
    :(FTP_Logout(vFTP_ID)#0)
    Else
        $OK:=True  `Toutes les commandes ont été exécutées sans erreur
End case
```

Note : Pour plus d'informations sur cet emploi particulier de la structure **Au cas ou**, reportez-vous à l'**Annexe A, Conseils de programmation**.

Exemple 2

```
$erreur:=FTP_Send(vFTP_ID;"";"")
```

FTP_SetPassive

FTP_SetPassive (ftp_ID ; modePassif) -> Résultat

Paramètre	Type		Description
ftp_ID	Entier long	➡	Référence d'une connexion FTP
modePassif	Entier	➡	0 = mode actif, 1 = mode passif (mode par défaut)
Résultat	Entier	⮌	Code d'erreur

Description

La commande *FTP_SetPassive* permet de définir les modalités d'échange sur le canal de transfert des données entre un serveur et un client FTP, lors de l'utilisation des commandes telles que *FTP_GetDirList*, *FTP_Send*, *FTP_Append* et *FTP_Receive*. Le mode de transfert spécifié est utilisé par ces commandes dès que vous avez exécuté *FTP_SetPassive*.

Les échanges entre un serveur et un client FTP se décomposent en deux parties : échanges sur le canal de contrôle (port 21 par défaut) et échanges sur le canal de transfert de données (port 20 par défaut). Généralement, les serveurs FTP sont dits “actifs”, car ils prennent en charge la gestion du transfert sur le canal de données et se connectent au client FTP.
Pour des raisons historiques, les commandes Internet de 4D demandent aux serveurs FTP de fonctionner en mode passif, ce qui consiste notamment à exécuter systématiquement la commande FTP “PASV” avant chaque échange sur le canal de transfert de données.
Toutefois, certains serveurs FTP ne permettent pas l'emploi du mode passif. En outre, ce mode peut être interdit par les systèmes de protection (firewall) des sites. Dans ce cas, la commande *FTP_SetPassive* vous permet de définir le mode actif pour le transfert des données.

Note : Il est recommandé de vérifier avec l'administrateur réseau le mode de transfert préconisé sur le site.

ftp_ID est l'identifiant de la session FTP établie avec *FTP_Login*.

Le paramètre *modePassif* permet de préciser le mode de transfert sur le canal de transfert des données :

- Si vous passez 0, vous demandez au serveur FTP de fonctionner en mode actif.
- Si vous passez 1, vous demandez au serveur FTP de fonctionner en mode passif (valeur par défaut).

FTP_SetType

FTP_SetType (ftp_ID ; ftp_Mode) -> Résultat

Paramètre	Type		Description
ftp_ID	Entier long	→	Référence d'une connexion FTP
ftp_Mode	Chaîne	→	"A" = Ascii ; "I" = [défaut] Image ; "L 8" = 8 bits Logique
Résultat	Entier	↩	Code d'erreur

Description

La commande *FTP_SetType* permet de modifier le mode de transfert FTP utilisé pendant les opérations d'envoi/réception. En général, il n'est pas nécessaire de modifier ce paramètre.

Toutefois, en raison des différences existant entre les diverses plates-formes et versions de FTP, le changement de mode de transfert peut, dans certains cas, s'avérer utile. Par exemple, le transfert de documents en texte seul peut nécessiter le mode Ascii.

ftp_ID est l'identifiant de la session FTP établie avec *FTP_Login*.

Passez dans le paramètre *ftp_Mode* le code du mode de transfert à utiliser. Ce paramétrage est pris en compte pour toutes les opérations d'envoi/réception ultérieures. Par défaut, le mode Image ("I") est utilisé.

FTP_System

FTP_System (ftp_ID ; infoSystème) -> Résultat

Paramètre	Type		Description
ftp_ID	Entier long	➡	Référence d'une connexion FTP
infoSystème	Chaîne	⬅	Informations sur le système
Résultat	Entier	➲	Code d'erreur

Description

La commande *FTP_System* retourne dans le paramètre *infoSystème* des informations décrivant le logiciel du serveur FTP.

ftp_ID est l'identifiant de la session FTP établie avec *FTP_Login*.

FTP_VerifyID

FTP_VerifyID (ftp_ID) -> Résultat

Paramètre	Type		Description
ftp_ID	Entier long	→	Référence d'une connexion FTP
		←	0 = La connexion est déjà fermée
Résultat	Entier	↩	Code d'erreur

Description

Un serveur FTP déconnecte automatiquement les comptes demeurés inactifs pendant une certaine période, déterminée par son administrateur (ce mécanisme est appelé timeout).
Chaque commande en interaction avec le serveur FTP provoque la remise à zéro du compteur d'inactivité. La commande *FTP_VerifyID* provoque également la remise à zéro du compteur d'inactivité de la session FTP spécifiée. Elle permet de conserver une session active en cas de risque de dépassement du délai.

Lors de son exécution, la commande *FTP_VerifyID* vérifie que la connexion n'a pas déjà été close. Si la session est toujours ouverte, la commande indique au serveur FTP de remettre à zéro le minuteur d'inactivité pour la session. Si la connexion a déjà été fermée, *FTP_VerifyID* retourne l'erreur appropriée, libère la mémoire utilisée par la session FTP et renvoie 0 dans le paramètre *ftp_ID*.

ftp_ID est l'identifiant de la session FTP établie avec *FTP_Login*.

FTP_Progress

FTP_Progress (gauche ; haut ; titreFenêtre ; texteThermo ; annuler) -> Résultat

Paramètre	Type		Description
gauche	Entier	➡	Coordonnée gauche de la fenêtre
haut	Entier	➡	Coordonnée supérieure de la fenêtre
titreFenêtre	Chaîne	➡	Titre de la fenêtre du thermomètre
texteThermo	Chaîne	➡	Texte au-dessus du thermomètre
annuler	Chaîne	➡	Libellé du bouton Annuler
Résultat	Entier	↩	Code d'erreur

Commande désactivée

Cette commande n'est plus prise en charge dans 4D Internet Commands à compter de la version v16 R2 et doit être supprimée de votre code. Si elle est appelée, elle retourne l'erreur -2201 (*fonction non implémentée*).

IC IMAP Review Mail

- Commandes IMAP4, Introduction
- IMAP_Capability
- IMAP_CloseCurrentMB
- IMAP_CopyToMB
- IMAP_CreateMB
- IMAP_Delete
- IMAP_DeleteMB
- IMAP_Download
- IMAP_GetCurrentMB
- IMAP_GetFlags
- IMAP_GetMBStatus
- IMAP_GetMessage
- IMAP_GetPrefs
- IMAP_ListMBs
- IMAP_Login
- IMAP_Logout
- IMAP_MsgFetch
- IMAP_MsgInfo
- IMAP_MsgLst
- IMAP_MsgLstInfo
- IMAP_MsgNumToUID
- IMAP_RenameMB
- IMAP_Search
- IMAP_SetCurrentMB
- IMAP_SetFlags
- IMAP_SetPrefs
- IMAP_SubscribeMB
- IMAP_UIDToMsgNum
- IMAP_VerifyID

Commandes IMAP4, Introduction

L'ensemble de commandes IMAP4 permet à vos bases de données d'accéder à des messages électroniques stockés sur un serveur de messagerie IMAP, de les traiter et de les récupérer. Les commandes IMAP sont conformes au protocole Internet Message Access Protocol, Version 4 révision 1 (IMAP4 rev1), défini par la rfc 2060. IMAP4 rev1 permet de gérer des dossiers de messages à distance, appelés "boîtes aux lettres", de la même manière que les boîtes aux lettres locales.

Les commandes IMAP permettent d'effectuer des opérations telles que la création, la suppression ou le changement de nom de boîtes aux lettres, la vérification de la présence de nouveaux messages, la suppression définitive de messages, la pose et la suppression de marqueurs (flags), la recherche de messages ou encore la récupération de parties de messages.

Terminologie

- **Connexion** : une "Connexion" désigne la totalité de la séquence d'interactions client/serveur IMAP, depuis la première connexion réseau (*IMAP_Login*) jusqu'à la fin de la connexion (*IMAP_Logout*).
- **Session** : une "Session" désigne la séquence d'interactions client/serveur IMAP à compter de la sélection d'une boîte aux lettres (*IMAP_SetCurrentMB*) jusqu'à la fin de cette sélection (*IMAP_SetCurrentMB*, *IMAP_CloseCurrentMB*) ou jusqu'à la fin de la connexion (*IMAP_Logout*).

Présentation des connexions IMAP

- Initialisation de la communication TCP : *IT_MacTCPInit* (dans le cas d'une connexion PPP, la commande *IT_PPPConnect* doit être exécutée au préalable).
- Ouverture d'une connexion : *IMAP_Login*.
- Gestion des boîtes aux lettres : liste, création, suppression, changement de nom, abonnement/désabonnement, statut des paramètres.
- Ouverture d'une session via la définition de la boîte aux lettres courante : *IMAP_SetCurrentMB*. Une fois que la boîte aux lettres courante est définie, vous pouvez gérer les messages qu'elle contient.
- Gestion des messages : liste, téléchargement ou suppression des messages ; liste des marqueurs de messages ; modification des marqueurs de messages ; copie vers une autre boîte aux lettres ; recherches et récupération de parties de messages sans téléchargement, etc.
- Une fois que vous avez fini de travailler avec les messages de la boîte aux lettres courante, vous pouvez fermer la session ou en ouvrir une autre en définissant une nouvelle boîte aux lettres courante. Dans tous les cas, le serveur IMAP mettra en permanence ses messages à jour. Par exemple, il supprimera tous les messages comportant le marqueur **\Deleted**.
- Lorsque vous avez terminé, vous devez vous déconnecter. Fermeture d'une connexion : *IMAP_Logout*.
- Autres opérations : définition des Préférences, capacité, vérification de la connexion et remise à zéro des compteurs d'inactivité avant déconnexion automatique sur le serveur IMAP.

Thèmes de commandes IMAP

Les commandes de traitement des messages sont réparties en deux thèmes : **IC IMAP Review Mail** (échanges avec le serveur IMAP) et **IC Downloaded Mail** (traitement en local), correspondant aux deux modes de lecture des messages électroniques. Lorsque vous lisez un message électronique depuis un serveur IMAP, les messages (ou les informations des messages) peuvent être importés dans des objets 4D (variables, champs, tableaux) ou téléchargés sur disque. Cette section décrit les possibilités offertes par les Commandes Internet de 4D concernant la lecture des messages depuis un serveur IMAP.

La taille des fichiers à télécharger va déterminer l'utilisation d'un mode par rapport à l'autre. Par exemple, un seul message électronique auquel est joint un fichier de 5 Mo pourrait facilement dépasser la capacité de stockage de la base de données. Seul un BLOB ou une image 4D est capable d'accueillir des données de cette taille, mais la conversion d'un message ou d'un document joint dans ce format est souvent inefficace car la messagerie cliente doit mobiliser de grandes ressources mémoire pour accéder à l'image ou au BLOB. Pour résoudre ce problème, la commande *IMAP_Download* transfère un message du serveur IMAP directement sur le disque local de l'utilisateur. Il suffit ensuite d'utiliser les commandes du thème **IC Downloaded Mail** pour manipuler le fichier sur disque.

Mécanismes des boîtes aux lettres

Une boîte aux lettres IMAP peut être considérée comme un dossier ; celui-ci peut contenir des fichiers et des sous-dossiers. De même, une boîte aux lettres peut contenir des messages et des sous-boîtes aux lettres.

L'accès à une boîte aux lettres s'effectue via son chemin d'accès hiérarchique complet. Les niveaux hiérarchiques sont désignés à l'aide d'un caractère séparateur, dépendant du serveur IMAP (la commande *IMAP_ListMBs* retourne le séparateur).

Vous pouvez utiliser le séparateur pour créer des boîtes aux lettres "filles" ou pour descendre ou remonter à l'intérieur de la hiérarchie. Tous les "enfants" d'un élément principal utilisent le même caractère séparateur.

Note : Vous ne pouvez gérer des messages qu'à partir du moment où une boîte aux lettres est définie comme boîte aux lettres courante (lorsqu'une session est ouverte) à l'aide de la commande *IMAP_SetCurrentMB*.

Chaque compte peut disposer d'une ou plusieurs boîtes aux lettres.
Les noms de boîtes aux lettres tiennent compte de la casse des caractères ; par conséquent, vous ne pouvez pas créer deux boîtes aux lettres dont seule la casse du nom diffère.

La boîte aux lettres INBOX est un cas particulier : elle existe dans chaque compte et est utilisée pour stocker les messages reçus. La boîte aux lettres INBOX est automatiquement créée à l'ouverture d'un compte. Un utilisateur ne peut pas supprimer la boîte aux lettres INBOX mais il peut la renommer. Dans ce cas, une nouvelle boîte aux lettres INBOX vide est immédiatement créée. Le nom INBOX ne tient pas compte de la casse.

Certaines informations, telles que le nombre total de messages ou de nouveaux messages, peuvent être obtenues même si la boîte aux lettres interrogée n'est pas la boîte aux lettres courante.

numéroMsg et uniqueID

L'utilisation des commandes IMAP nécessite une bonne compréhension des paramètres *numéroMsg* et *uniqueID*.

numéroMsg représente le numéro d'un message dans la boîte aux lettres au moment de l'exécution de la commande *IMAP_SetCurrentMB*.

Une fois qu'une boîte aux lettres courante est sélectionnée, les messages qu'elle contient sont numérotés de 1 à x (x étant le nombre d'éléments présents dans la boîte aux lettres). Les numéros sont affectés en fonction de l'ordre dans lequel les messages ont été reçus, le numéro 1 étant le plus ancien. Les numéros affectés aux messages ne sont valides qu'entre le moment où une boîte aux lettres courante a été sélectionnée (*IMAP_SetCurrentMB*) et sa fermeture (*IMAP_CloseCurrentMB*, *IMAP_SetCurrentMB* ou *IMAP_Logout*).

Au moment de la fermeture de la boîte aux lettres courante, tout message marqué comme "devant être supprimé" est effacé. Lorsque l'utilisateur se reconnecte au serveur IMAP, les messages présents dans la boîte aux lettres sont de nouveau numérotés de 1 à x. Par exemple, s'il y a 10 messages dans la boîte aux lettres et si les messages numérotés de 1 à 5 sont supprimés, les messages 6 à 10 seront renumérotés de 1 à 5 la prochaine fois que l'utilisateur consultera sa boîte aux lettres.

Pour illustrer ce fonctionnement, supposons que vous vous connectiez à un serveur IMAP et obteniez la liste de messages suivante :

numéroMsg	uniqueID	Date	De	Objet
1	10005	1 Jul 2001...	danw@acme.com	Clients potentiels...
2	10008	1 Jul 2001...	frank@acme.com	Commande de licence sur site
3	10012	3 Jul 2001...	joe@acme.com	Qui veut déjeuner ?
4	20000	4 Jul 2002...	kelly@acme.com	Appel de votre femme...
5	20001	4 Jul 2002...	track@fedex.com	Suivi FedEx

Pendant la session, vous supprimez les messages 3 et 4. Lorsque vous quittez la session, vos demandes de suppression sont exécutées. Lorsque vous retournez sur le serveur, la liste de messages est alors renumérotée ainsi :

numéroMsg	uniqueID	Date	De	Objet
1	10005	1 Jul 2001 ...	danw@acme.com	Clients potentiels...
2	10008	1 Jul 2001 ...	frank@acme.com	Commande de licence sur site
3	20001	4 Jul 2002 ...	track@fedex.com	Suivi FedEx

numéroMsg n'est pas une valeur statique se rapportant à un message spécifique, elle indique la position relative d'un message de la boîte aux lettres au moment de la sélection d'une boîte aux lettres courante. En revanche, *uniqueID* est un numéro unique affecté au message lors de sa réception par le serveur IMAP, dans un ordre strictement croissant. Chaque nouveau message reçoit un numéro *uniqueID* supérieur à ceux présents dans la boîte aux lettres.

Malheureusement, les serveurs IMAP n'utilisent pas *uniqueID* comme référence principale des messages. Aussi, lorsque vous manipulez des messages avec les commandes IMAP, vous devez passer *numéroMsg* comme paramètre d'identification des messages sur le serveur. Par conséquent, vous devez être prudent lorsque vous développez des applications qui référencent des messages dans la base de données tout en laissant le contenu du message sur le serveur.

Recommandations

Comme la caractéristique principale d'IMAP est l'interopérabilité et que celle-ci ne peut être contrôlée qu'en situation réelle, la recommandation principale est "Testez TOUT". Il est conseillé de tester vos postes clients avec tous les serveurs sur lesquels ils ont un compte.

Pour plus d'informations, vous pouvez consulter les sites suivants :

- IMAP Products and Services: *http://www.imap.org/products.html*
- MailConnect: *http://www.imc.org/imc-mailconnect*

Comparaison des commandes POP3 et IMAP4

Login	Equivalent	Pas de paramètre POP pour IMAP
VerifyID	Equivalent	
Delete	Equivalent	Les commandes IMAP suppriment en temps réel. Les commandes POP3 requièrent POP3_Logout pour supprimer les messages. IMAP_SetFlags positionnant le marqueur \Deleted permet d'obtenir le même résultat que POP3_Delete
Logout	Equivalent	
SetPrefs	Equivalent	Pas de *dossierDocsJoints* pour IMAP, le paramètre POP3 *dossierDocsJoints* est devenu optionnel
GetPrefs	Equivalent	Voir note *dossierDocsJoints* précédente
MsgLstInfo	Equivalent	
MsgInfo	Equivalent	
MsgLst	Equivalent	
UIDToMsgNum	Equivalent	*uniqueID* est un Entier long pour IMAP et un Alpha pour POP3
Download	Equivalent	
POP3_Reset	Pas d'équivalence directe	Nécessite la combinaison de IMAP_Search sur les marqueurs \Deleted et de IMAP_SetFlags pour supprimer les marqueurs \Deleted
POP3_BoxInfo	Pas d'équivalence directe	Nécessite la combinaison des commandes IMAP_SetCurrentMB et IMAP_MsgLstInfo
IMAP_ MsgNumToUID	Pas d'équivalence directe	
GetMessage	Presque équivalent	IMAP est plus puissant car il permet de sélectionner uniquement le corps du message
POP3_Charset	Pas d'équivalence	IMAP gère automatiquement les jeux de caractères
IMAP_Capability	Pas d'équivalence	Spécifique au protocole IMAP
IMAP_ListMBs	Pas d'équivalence	Spécifique au protocole IMAP
IMAP_GetMBStatus	Pas d'équivalence	Spécifique au protocole IMAP
IMAP_SetCurrentMB	Pas d'équivalence	Spécifique au protocole IMAP
IMAP_GetCurrentMB	Pas d'équivalence	Spécifique au protocole IMAP
IMAP_CloseCurrentMB	Pas d'équivalence	Spécifique au protocole IMAP
IMAP_CopyToMB	Pas d'équivalence	Spécifique au protocole IMAP
IMAP_SubscribeMB	Pas d'équivalence	Spécifique au protocole IMAP
IMAP_CreateMB	Pas d'équivalence	Spécifique au protocole IMAP
IMAP_DeleteMB	Pas d'équivalence	Spécifique au protocole IMAP
IMAP_RenameMB	Pas d'équivalence	Spécifique au protocole IMAP
IMAP_SetFlags	Pas d'équivalence	Spécifique au protocole IMAP
IMAP_GetFlags	Pas d'équivalence	Spécifique au protocole IMAP
IMAP_Search	Pas d'équivalence	Spécifique au protocole IMAP
IMAP_MsgFetch	Pas d'équivalence	Spécifique au protocole IMAP

Notes :

- Serveurs IMAP et POP3 : dans le cas des serveurs IMAP, ne gérez pas le paramètre *uniqueID* de la même manière car *uniqueID* est un entier long.

- La suppression ne fonctionne pas exactement de la même manière entre les protocoles POP3 et IMAP. *IMAP_Delete* supprime immédiatement les messages. Pour obtenir le même résultat que *POP3_Delete*, utilisez la commande *IMAP_SetFlags* et fixer le marqueur **\Deleted**. Pour obtenir le même résultat que *POP3_Reset*, utilisez la commande *IMAP_SetFlags* pour récupérer les marqueurs **\Deleted**.
- Pour une plus grande souplesse, les commandes Internet de 4D permettent de passer directement une référence de connexion IMAP aux commandes TCP de bas niveau et inversement. Pour plus d'informations, reportez-vous à la section **Routines de bas niveau, Présentation**.

IMAP_Capability

IMAP_Capability (imap_ID ; compatibilités) -> Résultat

Paramètre	Type		Description
imap_ID	Entier long	⇒	Référence de connexion IMAP
compatibilités	Texte	⇐	Compatibilités IMAP
Résultat	Entier	⇒	Code d'erreur

Description

La commande *IMAP_Capability* retourne une valeur Texte contenant la liste des différentes compatibilités du serveur IMAP. Les éléments de la liste sont séparés par des espaces.

La liste des compatibilités indique la version d'IMAP ainsi que les fonctions optionnelles (extensions, révisions ou amendements au protocole IMAP4rev1) acceptées par le serveur.

La compatibilité IMAP4rev1 doit apparaître dans le texte afin d'assurer le bon fonctionnement du serveur avec 4D Internet Commands.

IMAP_CloseCurrentMB

IMAP_CloseCurrentMB (imap_ID) -> Résultat

Paramètre	Type		Description
imap_ID	Entier long	➡	Référence de connexion IMAP
Résultat	Entier	⮌	Code d'erreur

Description

La commande *IMAP_CloseCurrentMB* permet de refermer la boîte aux lettres courante sans devoir en sélectionner une autre ni exécuter *IMAP_Logout*. *IMAP_CloseCurrentMB* provoque la suppression définitive de tous les messages comportant le marqueur **\Deleted**.

Note : IMAP permet aux utilisateurs de travailler simultanément avec la même boîte aux lettres en mode client/serveur. Lorsqu'un utilisateur effectue une synchronisation et conserve la connexion ouverte, la dernière boîte aux lettres utilisée reste sélectionnée. Si un autre utilisateur tente d'utiliser la même boîte aux lettres, il obtiendra des informations obsolètes ou ne pourra pas travailler correctement (en fonction du paramétrage du serveur).

imap_ID contient la référence d'une session ouverte avec *IMAP_Login*

IMAP_CopyToMB

IMAP_CopyToMB (imap_ID ; premierMsg ; dernierMsg ; nomBLCible ; supprMsg) -> Résultat

Paramètre	Type		Description
imap_ID	Entier long	➡	Référence de connexion IMAP
premierMsg	Entier long	➡	Numéro du premier message
dernierMsg	Entier long	➡	Numéro du dernier message
nomBLCible	Texte	➡	Nom de la boîte aux lettres de destination
supprMsg	Entier	➡	0 = Ne pas supprimer de la boîte aux lettres source, 1 = Supprimer de la boîte aux lettres source
Résultat	Entier	↩	Code d'erreur

Description

La commande *IMAP_CopyToMB* recopie le(s) message(s) appartenant à l'intervalle *premierMsg;dernierMsg* à la suite des messages de la boîte aux lettres de destination *nomBLCible*. Les marqueurs et dates internes des messages sont généralement conservés dans la boîte aux lettres de destination, toutefois ce fonctionnement dépend de l'implémentation du serveur IMAP.

Après la copie, les messages originaux ne sont pas supprimés de la boîte aux lettres source. Si vous voulez les supprimer, vous disposez de trois solutions :

- utiliser la commande *IMAP_Delete*,
- passer la valeur 1 dans le paramètre optionnel *supprMsg*,
- placer le marqueur *IMAP_SetFlags* (**\Deleted**) : les messages seront supprimés à la fermeture de la session.

Note : Le paramètre *supprMsg* force l'exécution de *IMAP_Delete* ; par conséquent, la suppression concernera les messages situés dans l'intervalle *premierMsg;dernierMsg* ainsi que TOUS les messages comportant le marqueur **\Deleted**.

Si la boîte aux lettres de destination n'existe pas, une erreur est retournée.

imap_ID contient la référence d'une session ouverte avec *IMAP_Login*.

Le paramètre *premierMsg* désigne le numéro du premier message à examiner. Ce numéro représente la position d'un message dans la liste de tous les messages de la boîte aux lettres courante.
Le paramètre *dernierMsg* indique le numéro du dernier message à examiner. Ce numéro représente la position d'un message dans la liste de tous les messages de la boîte aux lettres courante.

Note : Si le paramètre *premierMsg* est supérieur au paramètre *dernierMsg*, les commandes *IMAP_Delete*, *IMAP_MsgLstInfo*, *IMAP_MsgLst*, *IMAP_SetFlags*, *IMAP_GetFlags* et *IMAP_CopyToMB* ne retournent pas d'erreur et ne font rien.

nomBLCible contient le nom et le chemin d'accès de la boîte aux lettres devant recevoir les messages copiés.

Le paramètre optionnel *supprMsg* vous permet d'indiquer si vous souhaitez que les messages copiés soient supprimés de la boîte aux lettres source :

- 0= Ne pas supprimer les messages (valeur par défaut) ;
- 1= Supprimer les messages.

Si *supprMsg* est omis, la valeur par défaut est utilisée.

Si la copie échoue, le message n'est pas supprimé de la boîte aux lettres source.
Si l'utilisateur ne dispose pas d'accès suffisants pour supprimer des messages, une erreur est générée.

IMAP_CreateMB

IMAP_CreateMB (imap_ID ; nomBL) -> Résultat

Paramètre	Type		Description
imap_ID	Entier long	➡	Référence de connexion IMAP
nomBL	Texte	➡	Nom de la boîte aux lettres à créer
Résultat	Entier	⮌	Code d'erreur

Description

La commande *IMAP_CreateMB* crée une boîte aux lettres avec le nom spécifié par *nomBL*. Si le séparateur hiérarchique du serveur IMAP apparaît dans ce paramètre, le serveur IMAP créera les boîtes aux lettres parentes nécessaires à la création de la boîte spécifiée.

Par exemple, si la création de la boîte "Projets/IMAP/Doc" est demandée sur un serveur dont le séparateur hiérarchiques est "/" :

- Seule la boîte aux lettres "Doc" est créée si "Projets" et "IMAP" existent déjà.
- Les boîtes "IMAP" et "Doc" sont créées si "Projets" existe déjà.
- Les boîtes aux lettres "Projets", "IMAP" et "Doc" sont créées si elles n'existent pas déjà.

imap_ID contient la référence d'une session ouverte avec *IMAP_Login*.

nomBL contient le nom complet de la boîte aux lettres à créer (pour plus d'informations sur les noms de boîtes, reportez-vous à l'introduction de la section IMAP).

Note : Toute tentative de création d'un boîte aux lettres nommée INBOX (nom réservé signifiant "boîtes aux lettres primaire pour cet utilisateur sur ce serveur") ou déjà existante provoquera une erreur.

IMAP_Delete

IMAP_Delete (imap_ID ; premierMsg ; dernierMsg) -> Résultat

Paramètre	Type		Description
imap_ID	Entier long	➡	Référence de connexion IMAP
premierMsg	Entier long	➡	Numéro du premier message
dernierMsg	Entier long	➡	Numéro du dernier message
Résultat	Entier	⮌	Code d'erreur

Description

La commande *IMAP_Delete* place le marqueur **\Deleted** dans chaque message appartenant à l'intervalle compris entre *premierMsg* et *dernierMsg*, puis supprime tous les messages comportant ce marqueur (y compris ceux pour lesquels le marqueur **\Deleted** avait été placé préalablement au cours de la session). La suppression est effectuée par le serveur IMAP lorsque la connexion est close (*IMAP_Logout*), lorsqu'une autre boîte aux lettres est sélectionnée (*IMAP_SetCurrentMB*) ou lorsque la boîte aux lettres courante est refermée (*IMAP_CloseCurrentMB*).

Si vous ne voulez pas que la suppression soit immédiate, vous pouvez également placer les marqueurs **\Delete** à l'aide de la commande *IMAP_SetFlags* et ainsi provoquer une suppression différée des messages.

imap_ID contient la référence d'une session ouverte avec *IMAP_Login*.

Le paramètre *premierMsg* désigne le numéro du premier message à supprimer.

Le paramètre *dernierMsg* désigne le numéro du dernier message à supprimer.

Note : Si le paramètre *premierMsg* est supérieur au paramètre *dernierMsg*, les commandes *IMAP_Delete*, *IMAP_MsgLstInfo*, *IMAP_MsgLst*, *IMAP_SetFlags*, *IMAP_GetFlags* et *IMAP_CopyToMB* ne retournent pas d'erreur et ne font rien.

IMAP_DeleteMB

IMAP_DeleteMB (imap_ID ; nomBL) -> Résultat

Paramètre	Type		Description
imap_ID	Entier long	⇒	Référence de connexion IMAP
nomBL	Texte	⇒	Nom de la boîte aux lettres à supprimer
Résultat	Entier	⇐	Code d'erreur

Description

La commande *IMAP_DeleteMB* supprime définitivement la boîte aux lettres dont le nom est spécifié par *nomBL*. Si vous tentez de supprimer une boîte aux lettres INBOX ou une boîte aux lettres qui n'existe pas, une erreur est générée.

La commande *IMAP_DeleteMB* ne peut pas supprimer de boîte aux lettres contenant des sous-boîtes et comportant l'attribut **\Noselect**.

Il est toutefois possible de supprimer une boîte aux lettres contenant des sous-boîtes mais ne comportant pas l'attribut **\Noselect**. Dans ce cas, tous les messages de la boîte sont supprimés et l'attribut **\Noselect** lui est automatiquement appliqué.

Note : Le protocole IMAP ne garantit pas la possibilité de supprimer une boîte aux lettres non vide, bien que certains serveurs le permettent. Si vous autorisez ce fonctionnement, vous devez prévoir des solutions alternatives en cas d'échec.
Par ailleurs, vous ne devez pas autoriser la suppression de la boîte aux lettres courante pendant qu'elle est ouverte — vous devez en premier lieu la refermer. Certains serveurs ne permettent pas la suppression de la boîte aux lettres courante.

imap_ID contient la référence d'une session ouverte avec *IMAP_Login*.

nomBL contient le nom complet de la boîte aux lettres à supprimer.

IMAP_Download

IMAP_Download (imap_ID ; numéroMsg ; enTêteSeul ; nomFichier ; majSeen) -> Résultat

Paramètre	Type		Description
imap_ID	Entier long	➡	Référence de connexion IMAP
numéroMsg	Entier long	➡	Numéro du message
enTêteSeul	Entier	➡	0 = Message entier, 1 = En-tête seul
nomFichier	Texte	➡	Nom de fichier local
		⬅	Nom de fichier local utilisé
majSeen	Entier	➡	0 = Mise à jour du marqueur \Seen, 1 = Ne pas le mettre à jour
Résultat	Entier	↩	Code d'erreur

Description

La commande *IMAP_Download* permet de télécharger un message d'un serveur IMAP en local sur disque. Tout message IMAP contenant des fichiers joints ou dont la taille est supérieure à 32 Ko devra être téléchargé avec cette commande. Les fichiers joints peuvent être extraits uniquement à partir d'un message préalablement téléchargé.

imap_ID contient la référence d'une session ouverte avec *IMAP_Login*.

Le paramètre *numéroMsg* désigne le message à examiner. Le *numéroMsg* représente la position du message dans la liste courante des messages. Attention, le *numéroMsg* d'un message n'est pas une valeur stable, il peut différer d'une session à l'autre.

Le paramètre *enTêteSeul* vous permet de spécifier si vous souhaitez récupérer la totalité du message ou uniquement les informations des en-têtes.

Le paramètre *nomFichier* désigne le nom et/ou l'emplacement du fichier dans lequel vous souhaitez enregistrer le message. Cette valeur peut être spécifiée de trois manières :

- ""= Enregistre le fichier dans le dossier défini par *IMAP_SetPrefs*, avec le nom "temp1" (si un fichier de ce nom existe déjà, les noms "temp2", "temp3", etc., sont essayés).
- "nomFichier" = Enregistre le fichier dans le dossier défini par *IMAP_SetPrefs*, avec le nom *nomFichier*.
- "Chemin:nomFichier" = Enregistre le fichier en utilisant le chemin spécifié par *nomFichier*.

Dans les deux premiers cas, en l'absence de dossier spécifié par *IMAP_SetPrefs*, le message est enregistré dans le dossier de la structure de la base de données (avec 4D monoposte) ou dans le dossier de 4D Client (avec 4D Server).

Après l'exécution de la commande, le nom final du fichier est retourné dans le paramètre *nomFichier*. Si vous tentez d'appeler *IMAP_Download* avec un *nomFichier* qui existe déjà dans le dossier de téléchargement, ce nom est incrémenté et le nom réellement enregistré sur disque est retourné.

Le paramètre optionnel *majSeen* vous permet d'indiquer si le marqueur **\Seen** ("message lu") doit être ajouté ou non aux marqueurs du message. Vous pouvez passer une des valeurs suivantes :

- 0 = Ajouter le marqueur **\Seen** (valeur par défaut)
- 1 = Ne pas ajouter le marqueur **\Seen**

Si vous omettez le paramètre *majSeen*, le marqueur **\Seen** est ajouté par défaut au message.

IMAP_GetCurrentMB

IMAP_GetCurrentMB (imap_ID ; nomBL) -> Résultat

Paramètre	Type		Description
imap_ID	Entier long	⇒	Référence de connexion IMAP
nomBL	Texte	⇐	Nom de la boîte aux lettres courante
Résultat	Entier	⇒	Code d'erreur

Description

La commande *IMAP_GetCurrentMB* retourne le nom de la boîte aux lettres courante.

imap_ID contient la référence d'une session ouverte avec *IMAP_Login*.

Le paramètre *nomBL* retourne le nom complet de la boîte aux lettres courante. Si *nomBL* retourne une chaîne vide, aucune boîte aux lettres courante n'est sélectionnée.

IMAP_GetFlags

IMAP_GetFlags (imap_ID ; premierMsg ; dernierMsg ; tabMarqMsg ; tabNumMsg) -> Résultat

Paramètre	Type		Description
imap_ID	Entier long	➡	Référence de connexion IMAP
premierMsg	Entier long	➡	Numéro du premier message
dernierMsg	Entier long	➡	Numéro du dernier message
tabMarqMsg	Tableau chaîne	⬅	Valeurs des marqueurs pour chaque message
tabNumMsg	Tableau entier long	⬅	Tableau des numéros de messages
Résultat	Entier	↩	Code d'erreur

Description

La commande *IMAP_GetFlags* retourne la liste des marqueurs pour les messages définis.

imap_ID contient la référence d'une session ouverte avec *IMAP_Login*

Le paramètre *premierMsg* désigne le numéro du premier message à examiner. Ce numéro représente la position d'un message dans la liste de tous les messages de la boîte aux lettres courante.

Le paramètre *dernierMsg* indique le numéro du dernier message à examiner. Ce numéro représente la position d'un message dans la liste de tous les messages de la boîte aux lettres courante.

Note : Si le paramètre *premierMsg* est supérieur au paramètre *dernierMsg*, les commandes *IMAP_Delete*, *IMAP_MsgLstInfo*, *IMAP_MsgLst*, *IMAP_SetFlags*, *IMAP_GetFlags* et *IMAP_CopyToMB* ne retournent pas d'erreur et ne font rien.

Le tableau *tabMarqMsg* reçoit la liste des marqueurs, séparés par des espaces, de chaque message dont le numéro est inclus entre *premierMsg* et *dernierMsg*.

Le tableau *tabNumMsg* retourne les numéros des messages compris entre *premierMsg* et *dernierMsg*.

IMAP_GetMBStatus

IMAP_GetMBStatus (imap_ID ; nomBL ; nbMsg ; nbNouvMsg ; nbMsgNonLus ; uniqueIDBL) -> Résultat

Paramètre	Type		Description
imap_ID	Entier long	→	Référence de connexion IMAP
nomBL	Texte	→	Nom de boîte aux lettres
nbMsg	Entier long	←	Nombre de messages dans la boîte aux lettres spécifiée
nbNouvMsg	Entier long	←	Nombre de messages avec le marqueur \Recent
nbMsgNonLus	Entier long	←	Nombre de messages sans le marqueur \Seen
uniqueIDBL	Entier long	←	Numéro d'identification unique de la boîte aux lettres spécifiée
Résultat	Entier	↩	Code d'erreur

Description

La commande *IMAP_GetMBStatus* retourne le statut des paramètres de la boîte aux lettres spécifiée par *nomBL*. Cette commande ne modifie ni la boîte aux lettres courante (cf. *IMAP_SetCurrentMB*), ni le statut des messages présents dans la boîte aux lettres (en particulier, la commande ne supprime généralement pas les marqueurs **\Recent**, toutefois ce fonctionnement peut varier en fonction des paramétrages du serveur IMAP4). Cette commande alternative permet d'obtenir le statut des paramètres sans désélectionner la boîte aux lettres courante.

IMAP_GetMBStatus est particulièrement utile pour :

- Tester ou récupérer le numéro d'identification unique d'une boîte aux lettres.
- Vérifier les messages récents ou non lus d'une boîte aux lettres sans ouvrir de session spécifique.

Important : Il est fortement recommandé de ne pas utiliser la commande *IMAP_GetMBStatus* avec la boîte aux lettres courante. Dans ce cas en effet, il y a risque de désynchronisation des informations (concernant en particulier les nouveaux messages).

imap_ID contient la référence d'une session ouverte avec *IMAP_Login*.

Le paramètre *nomBL* contient le nom complet de la boîte aux lettres dont vous souhaitez obtenir les paramètres de statut.

Note : A la différence de la commande *IMAP_ListMBs*, le paramètre *nomBL* n'accepte pas de caractères "Joker" avec *IMAP_GetMBStatus*.

Le paramètre *nbMsg* retourne le nombre de messages présents dans la boîte aux lettres (retourne -1 en cas d'erreur).

Le paramètre *nbNouvMsg* retourne le nombre de messages récents présents dans la boîte aux lettres (retourne -1 en cas d'erreur).

Le paramètre *nbMsgNonLus* retourne le nombre de messages non lus présents dans la boîte aux lettres (retourne -1 en cas d'erreur).

Le paramètre *uniqueIDBL* retourne le numéro d'identification unique de la boîte aux lettres (retourne -1 en cas d'erreur).
Avec le protocole IMAP4, le nom n'est pas suffisant pour identifier une boîte aux lettres. Pour cela, un numéro d'identification unique est attribué à chaque boîte aux lettres. Cet identifiant est particulièrement utile pour synchroniser des tâches.
Ainsi, vous pouvez vérifier si une boîte aux lettres "A" a été renommée "B" ou supprimée, simplement en vérifiant son numéro d'identification unique. De même, l'identifiant vous permet de savoir si une boîte aux lettres "A" a été supprimée puis remplacée par une nouvelle boîte nommée "A".

IMAP_GetMessage

IMAP_GetMessage (imap_ID ; numéroMsg ; décalage ; longueur ; partieMsg ; texteMsg {; majSeen}) -> Résultat

Paramètre	Type		Description
imap_ID	Entier long	⇒	Référence de connexion IMAP
numéroMsg	Entier long	⇒	Numéro du message
décalage	Entier long	⇒	Caractère à partir duquel commencer la récupération
longueur	Entier long	⇒	Nombre de caractères à renvoyer
partieMsg	Entier	⇒	0 = Message entier, 1 = En-tête uniquement, 2 = Corps uniquement
texteMsg	Texte	⇐	Texte du message
majSeen	Entier	⇒	0 = Mettre à jour le marqueur \Seen, 1 = Ne pas le mettre à jour
Résultat	Entier	↩	Code d'erreur

Description

La commande *IMAP_GetMessage* retourne le texte du message désigné par *numéroMsg* dans la boîte aux lettres courante définie par la commande *IMAP_SetCurrentMB*. Sauf spécification contraire de la commande *IMAP_SetPrefs*, les caractères Retour à la ligne (Line feed) à l'intérieur du message sont supprimés.
La commande *IMAP_GetMessage* peut retourner soit l'intégralité du message, y compris les informations des zones d'en-tête, soit uniquement une partie du message (en-tête ou corps) en fonction de la valeur du paramètre *partieMsg*.

imap_ID contient la référence d'une session ouverte avec *IMAP_Login*.

Le paramètre *numéroMsg* désigne le message à récupérer dans la boîte aux lettres. Le *numéroMsg* représente la position du message dans la liste courante de messages. Attention, le *numéroMsg* d'un message n'est pas une valeur stable, il diffère d'une session à l'autre.

Le paramètre *décalage* vous permet d'indiquer la position du caractère (calculée par rapport au début de la partie du message à récupérer) à partir duquel commencer la lecture. Dans la plupart des cas, vous passerez 0 (zéro) dans ce paramètre.

Le paramètre *longueur* indique le nombre de caractères à récupérer au-delà de la position de *décalage*.

Le paramètre *partieMsg* permet d'indiquer la partie du message à récupérer. Vous pouvez utiliser les valeurs 0, 1 ou 2 :

- 0 = Message entier
- 1 = En-tête uniquement
- 2 = Corps uniquement (c'est-à-dire à partir du premier Text/ plain rencontré).
  Lorsque vous récupérez le message entier ou l'en-tête uniquement, vous obtenez du texte brut non décodé. Lorsque vous récupérez le corps uniquement, le texte obtenu est automatiquement décodé et converti si nécessaire (reportez-vous à la description de la commande *POP3_Charset* pour plus d'informations sur les règles de conversion et de décodage).

Le paramètre optionnel *majSeen* vous permet d'indiquer si le marqueur **\Seen** ("message lu") doit être ajouté ou non aux marqueurs du message. Vous pouvez passer une des valeurs suivantes :

- 0 = Ajouter le marqueur \Seen (valeur par défaut)
- 1 = Ne pas ajouter le marqueur \Seen

Si vous omettez le paramètre *majSeen*, le marqueur **\Seen** est ajouté par défaut au message.

Le texte récupéré est retourné dans la variable *texteMsg*.

IMAP_GetPrefs

IMAP_GetPrefs (retoursLigne ; dossierMsg) -> Résultat

Paramètre	Type		Description
retoursLigne	Entier	⇐	0 = Ne pas retirer les retours à la ligne, 1 = Retirer les retours à la ligne
dossierMsg	Texte	⇐	Chemin d’accès au dossier des messages
Résultat	Entier	⇒	Code d'erreur

Description

La commande *IMAP_GetPrefs* vous permet de connaître les préférences courantes pour les commandes IMAP. Les préférences sont retournées dans les variables listées dans les paramètres.

Le paramètre *retoursLigne* retourne le paramétrage courant de l’option de suppression des retours à la ligne.

Le paramètre *dossierMsg* retourne le chemin d’accès local du dossier où sont enregistrés par défaut les messages récupérés.

IMAP_ListMBs

IMAP_ListMBs (imap_ID ; réfBL ; nomBL ; tabNomsBL ; tabAttribsBL ; tabHiérarBL ; abonnementsBL) -> Résultat

Paramètre	Type		Description
imap_ID	Entier long	➡	Référence de connexion IMAP
réfBL	Texte	➡	Chaîne vide ou Référence de boîte aux lettres ou Niveau hiérarchique
nomBL	Texte	➡	Chaîne vide ou Nom de boîte aux lettres ou Jokers
tabNomsBL	Tableau chaîne	⬅	Tableau de noms de boîtes aux lettres (chemins d'accès)
tabAttribsBL	Tableau chaîne	⬅	Tableau d'attributs de boîtes aux lettres
tabHiérarBL	Tableau chaîne	⬅	Tableau de séparateurs hiérarchiques
abonnementsBL	Entier	➡	0 = Lister toutes les boîtes aux lettres disponibles 1 = Lister uniquement les boîtes aux lettres abonnées
Résultat	Entier	↩	Code d'erreur

Description

La commande *IMAP_ListMBs* retourne la liste des boîtes aux lettres disponibles pour l'utilisateur connecté, ainsi que les informations liées. Si la commande échoue, les tableaux sont retournés vides.

réfBL et *nomBL* doivent être traités conjointement, car la liste des boîtes aux lettres résultante dépendra de la combinaison des valeurs de ces deux paramètres.

Si vous passez 1 dans le dernier paramètre, *abonnementsBL*, la liste retournée est restreinte aux boîtes aux lettres auxquelles l'utilisateur est abonné (voir la commande *IMAP_SubscribeMB*).
Lorsque l'exécution de *IMAP_ListMBs* prend beaucoup de temps — à cause d'un grand nombre de boîtes aux lettres à examiner, d'une structure hiérarchique complexe, etc. — vous pouvez :

- soit utiliser des jokers (voir ci-dessous),
- soit passer 1 dans le paramètre *abonnementsBL* pour lister uniquement les boîtes aux lettres définies via la commande *IMAP_SubscribeMB*.

imap_ID contient la référence d'une session ouverte avec *IMAP_Login*.

Le paramètre *réfBL* doit être combiné avec le paramètre *nomBL* afin de déterminer les boîtes aux lettres à utiliser. La référence de boîte aux lettres (*réfBL*) est l'équivalent d'un Répertoire de travail courant sur les systèmes Unix. En d'autres termes, le nom de la boîte aux lettres (*nomBL*) sera considéré comme un nom de fichier situé dans le répertoire désigné par la référence (*réfBL*). Attention, les spécifications IMAP indiquent que l'interprétation de la référence (*réfBL*) est "liée à l'implémentation" — ce qui signifie qu'elle n'est pas obligatoire. Il est donc fortement recommandé de prévoir un mode opératoire n'utilisant pas le paramètre *réfBL* afin de pouvoir dialoguer avec des serveurs plus anciens n'interprétant pas ce paramètre.

Si *réfBL* est une chaîne vide, seul le paramètre *nomBL* sera utilisé pour lister les boîtes aux lettres.
Si *réfBL* contient le nom d'une boîte aux lettres ou un niveau hiérarchique de boîtes aux lettres, il sera utilisé pour définir le contexte à partir duquel le paramètre nomBL sera interprété.

Note : Il est fortement recommandé de placer un séparateur hiérarchique à la fin du paramètre *réfBL* lorsqu'il est utilisé, afin d'assurer la compatibilité de la commande quel que soit le serveur IMAP utilisé.

La valeur à passer dans le paramètre *nomBL* dépend de celle du paramètre *réfBL*. Si *nomBL* est une chaîne vide, le séparateur hiérarchique est retourné.

Note : Si vous mettez en place un système de connexions multiples à l'aide du paramètre *réfBL*, vous devez permettre à l'utilisateur de placer ou non un séparateur hiérarchique au début du nom de la boîte aux lettres. En effet, la gestion du séparateur en début de nom varie d'un serveur à l'autre, voire entre deux gestionnaires de messagerie sur le même serveur. Dans certains cas, ce caractère signifiera "ne pas tenir compte du paramètre réfBL", dans d'autres cas les deux paramètres seront concaténés et le caractère séparateur sera ignoré.

Le tableau *tabNomsBL* reçoit la liste des noms de boîtes aux lettres disponibles.
Le tableau *tabAttribsBL* reçoit la liste des attributs des boîtes aux lettres disponibles.

Attributs des boîtes aux lettres
Quatre attributs sont disponibles :

- **\Noinferiors** : aucun niveau hiérarchique inférieur n'existe ni ne peut être créé.
- **\Noselect** : ce nom ne peut être utilisé en tant que boîte aux lettres sélectionnable.

- **\Marked** : le serveur a marqué la boîte aux lettres comme "intéressante" ; la boîte aux lettres contient probablement des messages nouveaux par rapport à sa dernière sélection.
- **\Unmarked** : la boîte aux lettres ne contient pas de messages nouveaux par rapport à sa dernière sélection.

Le tableau *tabHiérarBL* reçoit la liste des séparateurs hiérarchiques des boîtes aux lettres disponibles.
Le séparateur hiérarchique est le caractère utilisé pour délimiter les niveaux hiérarchiques dans un nom de boîte aux lettres. Vous pouvez utiliser ce caractère pour créer des boîtes aux lettres "filles" ou pour remonter ou descendre les niveaux hiérarchiques. Tous les sous-éléments d'un niveau hiérarchique principal utilisent le même caractère séparateur.

Le paramètre *abonnementsBL* vous permet d'indiquer si vous souhaitez ne récupérer que la liste des boîtes aux lettres auxquelles l'utilisateur est abonné : pour cela, passez 1 dans *abonnementsBL*.
Si vous passez 0 ou omettez ce paramètre, toutes les boîtes aux lettres disponibles sont listées.

Exemple 1

L'exemple suivant :

```
IMAP_ListMBs(imap_ID;"4DIC/Work/";"Test";tabNomsBL;tabAttribsBL;tabHiérarBL)
```

... retourne toutes les boîtes aux lettres disponibles depuis la boîte "4DIC/Work/Test".
Rappelons que si le serveur IMAP n'interprète pas les paramètres comme vous le souhaitez, n'utilisez pas le paramètre *réfBL* et concaténez les valeurs de *réfBL* et *nomBL* dans le paramètre *nomBL* :

```
IMAP_ListMBs(imap_ID;"";"4DIC/Work/Test";tabNomsBL;tabAttribsBL;tabHiérarBL)
```

Exemple 2

L'exemple suivant :

```
IMAP_ListMBs(imap_ID;"";"";tabNomsBL;tabAttribsBL;tabHiérarBL)
```

... retourne le séparateur hiérarchique.

Utiliser des jokers

Vous pouvez utiliser des jokers dans les paramètres *réfBL* et *nomBL* afin de faciliter la sélection de boîte aux lettres. Vous trouverez ci-dessous un exemple utilisant des jokers usuels, mais notez que l'interprétation des jokers dépend du serveur IMAP. Il est donc possible que ces exemples ne fonctionnent pas. Dans ce cas, vérifiez les jokers de votre serveur IMAP.

- " * " remplace tout caractère à son emplacement :

```
IMAP_ListMBs(imap_ID;"";"*";tabNomsBL;tabAttribsBL;tabHiérarBL)
```

... retourne toutes les boîtes aux lettres disponibles pour l'utilisateur connecté.

```
IMAP_ListMBs(imap_ID;"";"Suivi*";tabNomsBL;tabAttribsBL;tabHiérarBL)
```

... retourne toutes les boîtes aux lettres disponibles débutant par la racine "Suivi".

- " % " est semblable à " * ", mais ne remplace pas le séparateur hiérarchique. Si le joker " % " est le dernier caractère du paramètre *nomBL*, les niveaux hiérarchiques correspondants sont également retournés. Si ces niveaux hiérarchiques ne sont pas des boîtes aux lettres sélectionnables, ils sont retournés avec l'attribut **\Noselect** (voir paragraphe "Attributs des boîtes aux lettres") :

```
IMAP_ListMBs(imap_ID"";"Suivi/%";tabNomsBL;tabAttribsBL;tabHiérarBL)
```

... retourne toutes les boîtes aux lettres disponibles débutant par la racine "Suivi", plus un sous-niveau hiérarchique supplémentaire.

Le joker “%” peut être utile lors d’une analyse niveau par niveau de la hiérarchie des boîtes aux lettres. Imaginons la hiérarchie de boîtes aux lettres suivante :

```
INBOX
    BoîteA
        BoîteAA
        BoîteAB
    BoîteB
        BoîteBA
        BoîteBB
    BoîteC
        BoîteCA
        BoîteCB
```

```
IMAP_ListMBs(imap_ID;"";"%";tabNomsBL;tabAttribsBL;tabHiérarBL)
```

... retourne INBOX, BoîteA, BoîteB et BoîteC.

```
IMAP_ListMBs(imap_ID;"";"BoîteA%";tabNomsBL;tabAttribsBL;tabHiérarBL)
```

... retourne BoîteAA et BoîteAB.

A l’aide de cette technique, vous pouvez proposer à l’utilisateur une certaine flexibilité sans le noyer sous les informations pouvant être retournées par un appel du type

```
IMAP_ListMBs(imap_ID;"";"*";tabNomsBL;tabAttribsBL;tabHiérarBL).
```

Notez que les serveurs IMAP eux-mêmes peuvent limiter le nombre de niveaux à analyser.

IMAP_Login

IMAP_Login (nomServeur ; nomUtilisateur ; motDePasse ; imap_ID {; paramSession}) -> Résultat

Paramètre	Type		Description
nomServeur	Chaîne	➡	Nom ou adresse IP du serveur de courrier IMAP
nomUtilisateur	Chaîne	➡	Nom de l'utilisateur
motDePasse	Chaîne	➡	Mot de passe
imap_ID	Entier long	⬅	Référence de cette connexion IMAP
paramSession	Entier long	➡	1 = Utiliser SSL, 0 ou omis = Ne pas utiliser SSL
Résultat	Entier	↩	Code d'erreur

Description

La commande *IMAP_Login* connecte l'utilisateur défini par *nomUtilisateur* et *motDePasse* au serveur de courrier IMAP.

IMAP_Login retourne un numéro d'identification spécifique pour la connexion (*imap_ID*) que les commandes IMAP ultérieures devront utiliser.

La connexion peut être close via la commande *IMAP_Logout* ou si le compteur d'inactivité du serveur IMAP a atteint le délai de timeout.

Le paramètre *nomServeur* contient le nom ou l'adresse IP du serveur de courrier IMAP. Il est généralement conseillé d'utiliser le nom du serveur.

Le paramètre *nomUtilisateur* contient le nom de l'utilisateur du serveur de courrier IMAP. Ce nom ne doit pas contenir le nom du domaine. Par exemple, dans le cas de l'adresse "jack@4d.com", le *nomUtilisateur* est "jack".

Le paramètre *motDePasse* est le mot de passe correspondant au *nomUtilisateur* sur le serveur de courrier IMAP.

Le paramètre *imap_ID* est une variable de type Entier long dans laquelle la référence de la connexion venant d'être établie est renvoyée. Une variable 4D doit être passée à ce paramètre afin d'accepter les résultats retournés. Cette variable devra être utilisée par toutes les commandes suivantes effectuant des actions liées à cette connexion. Si la commande *IMAP_Login* échoue, *imap_ID* prend la valeur zéro.

Le paramètre optionnel *paramSession* vous permet d'activer le protocole SSL pour la connexion :

- si vous passez 1, la connexion au serveur IMAP sera effectuée en SSL (mode synchrone),
- si vous passez 0 ou omettez ce paramètre, la connexion sera effectuée en mode standard non sécurisé.

Exemple

Voici une séquence type de connexion :

```
$ErrorNum:=IMAP_Login(vHost;vUserName;vUserPassword;vImap_ID;1)
If($ErrorNum =0)
   C_TEXT(vCapability)
   $ErrorNum:=IMAP_Capability(vImap_ID;vCapability))
  ` Commandes IMAP utilisant le paramètre vImap_ID
End if
$ErrorNum:=IMAP_Logout(vImap_ID)
```

IMAP_Logout

IMAP_Logout (imap_ID) -> Résultat

Paramètre	Type		Description
imap_ID	Entier long	➡	Référence de connexion IMAP
		⬅	0 = Connexion refermée
Résultat	Entier	↩	Code d'erreur

Description

La commande *IMAP_Logout* referme la connexion IMAP référencée par la variable *imap_ID*.
Si la déconnexion du serveur IMAP est correctement effectuée, la valeur 0 (zéro) est retournée dans *imap_ID*.

Note : La fermeture de la connexion entraîne automatiquement celle de la session courante.

IMAP_MsgFetch

IMAP_MsgFetch (imap_ID ; numéroMsg ; donnéesMsg ; valeursMsg) -> Résultat

Paramètre	Type		Description
imap_ID	Entier long	⇒	Référence de connexion IMAP
numéroMsg	Entier long	⇒	Numéro du message
donnéesMsg	Texte	⇒	Elément(s) de données à récupérer
valeursMsg	Texte	⇐	Valeur(s) des données récupérées
Résultat	Entier	↩	Code d'erreur

Description

La commande *IMAP_MsgFetch* vous permet de récupérer un ou plusieurs éléments de données simples d'un message spécifique sans devoir le télécharger.

imap_ID contient la référence d'une session ouverte avec *IMAP_Login*.

Le paramètre *numéroMsg* désigne le message à examiner. Le *numéroMsg* représente la position du message dans la liste courante des messages. Attention, le *numéroMsg* d'un message n'est pas une valeur stable, il peut différer d'une session à l'autre.

Le paramètre *donnéesMsg* est une variable texte vous permettant de définir le ou les élément(s) de données que vous souhaitez récupérer. Si vous souhaitez définir plusieurs éléments de données, placez un espace entre chaque élément. Il existe deux sortes d'éléments de données :

- les *éléments de données simples*, récupérant une seule information,
- les *macro éléments de données*, récupérant plusieurs informations simples en une seule fois. Trois macros définissant des ensembles d'informations courants sont disponibles. Une macro doit être utilisée seule, c'est-à-dire sans autre macro ni élément de données.

Pour plus d'informations sur les éléments de données, reportez-vous plus loin aux paragraphes "Eléments de données simples" et "Macro éléments de données".

Le paramètre *valeursMsg* retourne soit une paire simple **Elément de données/Valeur de données**, soit une liste de paires **Elément de données/Valeur de données**.

- dans le premier cas, la structure du texte retourné est la suivante : **Elément de données+Espace+Valeur de données**
- dans le second cas, la structure du texte retourné est la suivante : **Elément de données1+Espace+Valeur de données1+Espace+Elément de données2+Espace+Valeur de données2**

valeursMsg peut contenir une liste entre parenthèses, une chaîne entre guillemets ou une chaîne simple, en fonction du paramètre *donnéesMsg*.

- Les listes entre parenthèses sont structurées de la manière suivante (voir l'exemple **FLAGS**) : (*Valeur de données1+Espace+Valeur de données2*). Des parenthèses vides sont retournées lorsqu'il n'y a aucune valeur de données. Cette règle ne s'applique pas aux listes d'adresses entre parenthèses (voir l'exemple **ENVELOPE**).
- Les chaînes entre guillemets sont structurées de la manière suivante (voir l'exemple **INTERNALDATE**) : *Elément de données+Guillemets+Valeur de données+Guillemets*. Lorsqu'une valeur de données est inexistante, une chaîne vide "" est retournée.
- Les chaînes simples (sans guillemets) indiquent des valeurs de type entier, entier long ou numérique et sont structurées de la manière suivante : *Elément de données+Espace+Valeur de données*. Dans ce cas, vous devrez généralement convertir les valeurs dans le type correspondant (voir l'exemple **UID**).

Note : Les guillemets sont utilisés lorsque les chaînes comportent des caractères spéciaux, tels que des espaces ou des parenthèses. Aussi, lors du traitement de la chaîne de caractères retournée par la commande **IMAP_Fetch**, les caractères guillemets sont pris en compte.

Eléments de données simples

- **INTERNALDATE**
  Récupère la date et l'heure internes du message sur le serveur IMAP. Il ne s'agit pas de la date et de l'heure de l'en-tête "Date", mais de la date et de l'heure indiquant le moment où le message a été reçu. Pour les messages envoyés via un serveur SMTP, cette information indique généralement la date

et l'heure de la livraison finale du message. Pour les messages envoyés après la commande **IMAP_Copy**, cette information indique généralement la date et l'heure internes du message source. L'élément de données **INTERNALDATE** retourne une chaîne entre guillemets.

Exemple :

```
données Msg:="INTERNALDATE"
$Err:=IMAP_MsgFetch(imap_ID;1;données Msg;valeursMsg)
```

valeursMsg retourne INTERNALDATE "17-Jul-2001 15:45:37 +0200"

- **FLAGS**
  Récupère entre parenthèses la liste des marqueurs définis pour le message. Les marqueurs sont séparés par des espaces.
  Exemple :

```
données Msg:="FLAGS"
$Err:=IMAP_MsgFetch(imap_ID;1;données Msg;valeursMsg)
```

valeursMsg retourne FLAGS () s'il n'y a aucun marqueur défini pour le message spécifié.
valeursMsg retourne FLAGS (\Seen \Answered) si les marqueurs **\Seen** et **\Answered** sont définis pour le message.

- **RFC822.SIZE**
  Récupère le nombre d'octets du message, exprimé dans le format RFC-822. Le nom de l'élément de données est séparé de la valeur retournée par un espace. Une chaîne sans guillemets est retournée, ce qui signifie que vous devrez probablement la convertir en entier long (voir l'exemple **UID**).
  Exemple :

```
données Msg:="RFC822.SIZE"
$Err:=IMAP_MsgFetch(imap_ID;1;données Msg;valeursMsg)
```

valeursMsg retourne RFC822.SIZE 99599

- **ENVELOPE**
  Récupère entre parenthèses la liste décrivant la partie d'en-tête du message. Le serveur traite cette partie en analysant les champs d'en-tête et en leur fixant des valeurs par défaut si nécessaire.
  Les champs d'en-tête sont retournés dans l'ordre suivant : date, objet, émetteur (from), expéditeur (sender), réponse à (reply-to), destinataire (to), copie (cc), copie discrète (bcc), en réponse à (in-reply-to) et message-id. Les champs date, objet, en réponse à et message-id sont des chaînes entre guillemets :
  ENVELOPE ("date" "objet" (émetteur) (expéditeur) (réponse à) (destinataire) (copie) (copie discrète) "en réponse à" "message-id")
  Exemple :

```
données Msg:="ENVELOPE"
$Err:=IMAP_MsgFetch(imap_ID;1;données Msg;valeursMsg)
```

valeursMsg retourne ENVELOPE ("Tue, 17 Jul 2001 17:26:34 +0200" "Test" (("RSmith" NIL "RSmith" "test")) (("RSmith" NIL "RSmith" "test")) (("RSmith" NIL "RSmith" "test")) (("RSmith" NIL "RSmith" "test")) () () "" "<ee6b33a.-1@Mail.x6foadRIbnm>")

Date :	"Tue, 17 Jul 2001 17:26:34 +0200"	En-tête date
Objet :	"Test"	En-tête objet
Emetteur :	(("RSmith" NIL "RSmith" "test"))	Structure d'adresse
Expéditeur :	(("RSmith" NIL "RSmith" "test"))	Structure d'adresse
Réponse à :	(("RSmith" NIL "RSmith" "test"))	Structure d'adresse
Destinataire :	(("RSmith" NIL "RSmith" "test"))	Structure d'adresse
Copie :	()	En-tête copie inutilisé
Copie discrète :	()	En-tête copie discrète inutilisé
En réponse à :	""	En-tête en réponse à inutilisé
Message-id :	"<ee6b33a.-1@Mail.x6foadRIbnm>"	En-tête message-id

Les en-têtes émetteur, expéditeur, réponse à, destinataire, copie et copie discrète sont des listes entre parenthèses de structures d'adresses. Une structure d'adresse est une liste décrivant une adresse de messagerie électronique. Les champs d'une structure d'adresse se présentent dans l'ordre suivant : nom, [SMTP] at-domain-list (source route), nom de la boîte aux lettres et nom de serveur. Par exemple, *(("RSmith" NIL "RSmith" "test"))*.

- **BODY**
  BODY retourne les mêmes informations que **BODYSTRUCTURE** excepté pour les données d'extension (voir **BODYSTRUCTURE**).
  Exemple :

```
données Msg:="BODY"
$Err:=IMAP_MsgFetch(imap_ID;1;données Msg;valeursMsg)
```

valeursMsg retourne BODY ("TEXT" "PLAIN" ("CHARSET" "us-ascii") NIL NIL "8BIT" 8 1)

- **BODYSTRUCTURE**
  Récupère la structure MIME du message. Le serveur traite cette partie en analysant les champs d'en-tête MIME dans l'en-tête et le corps du message. Cet élément de données est particulièrement utile pour analyser le contenu d'un message sans le télécharger. Par exemple, vous pouvez ainsi tester rapidement la taille de chaque partie ou le nom des fichiers joints. **BODYSTRUCTURE** retourne une liste entre parenthèses incluant des listes entre parenthèses, des chaînes entre guillemets et des chaînes sans guillemets.
  En fonction du contenu du message, **BODYSTRUCTURE** retournera soit une liste "non-multipart" soit une liste imbriquée ("multipart") :
  - Liste "**non-multipart**" entre parenthèses : semblable, par exemple, à un message électronique "non-multipart" ; la structure du corps d'un message texte simple de 48 lignes et 2279 octets peut être la suivante : *("TEXT" "PLAIN" ("CHARSET" "us-ascii") NIL NIL "8BIT" 8 1 NIL NIL NIL)*
    Les champs simples d'une liste "non-multipart" entre parenthèses apparaissent dans l'ordre suivant :

body type	Chaîne fournissant le type de contenu de media (Content-type: media type p. e. TEXT)
body subtype	Chaîne fournissant le sous-type de contenu de media (Content-type: sub type p. e. PLAIN)
body parameter	Liste entre parenthèses de paires attributs/valeurs
parenthesized list	[p. e. ("CHARSET" "US-ASCII" "NAME" "cc.diff") où "US-ASCII" est la valeur de "CHARSET" et "cc.diff" celle de "NAME".
body id	Chaîne fournissant le numéro d'ID du contenu (permet à un corps de faire référence à un autre). Ainsi, les corps de messages peuvent être référencés à l'aide du champ d'en-tête "Content-ID". Ce champ a une syntaxe particulière dans le cas d'un type de media multipart/alternative. Voir l'explication fournie dans la section de la RFC 2046 relative aux cas multipart/alternative.
body description	Chaîne décrivant le contenu
body encoding	Chaîne fournissant l'encodage de transfert du contenu (Content-Transfer-Encoding)
body size	Valeur numérique indiquant la taille du corps en octets. Notez que cette valeur concerne la taille pendant l'encodage de transfert et non la taille résultante après décodage.

    Un corps de type MESSAGE de sous-type RFC822 contient, juste après les champs simples, la structure d'enveloppe, la structure de corps et la taille en lignes de texte du message encapsulé.
    Un corps de type TEXT contient, juste après les champs simples, la taille en lignes de texte du corps. Notez que cette valeur concerne la taille pendant l'encodage et non la taille résultante après décodage.
    Les données d'extension suivent les champs simples et les champs de type listés ci-dessus. Ces données ne sont jamais retournées avec l'élément **BODY**, mais le sont avec **BODYSTRUCTURE**. Lorsqu'elles sont présentes, les données d'extension d'une liste "non-multipart" entre parenthèses doivent apparaître dans l'ordre suivant :

body MD5	Chaîne définissant la valeur MD5 du corps, comme défini dans [MD5]
body disposition	Liste entre parenthèses constituée d'une chaîne de type de disposition suivie d'une liste entre parenthèses de paires attributs/valeurs comme défini dans [DISPOSITION]
body language	Chaîne ou liste entre parenthèses indiquant le langage du corps comme défini dans [LANGUAGE-TAGS]

Exemple : *("TEXT" "PLAIN" ("CHARSET" "US-ASCII") NIL NIL "7BIT" 2279 48 NIL NIL NIL)*
Description : *("bodytype" "bodysubtype" (BodyParameterParenthesizedList) bodyId bodyDescription "bodyEncoding" BodySize BodySizeInTextLines ExtensionDataBODYmd5 ExtensionDataBodyDisposition ExtensionDataBodyLanguage)*

- Liste "**multipart**" entre parenthèses : c'est le cas de l'e-mail multipart ; elle inclut une liste "non-multipart" entre parenthèses.
  Les parenthèses imbriquées indiquent des parties multiples (multiple parts). Le premier élément de la liste entre parenthèses est un corps imbriqué et non un type de corps. Le deuxième élément de la liste est le sous-type multipart (mixed, digest, parallel, alternative, etc.).
  Le sous-type multipart est suivi des données d'extension.
  Lorsqu'elles sont présentes, les données d'extension doivent apparaître dans l'ordre suivant :

body parameter parenthesized list	Liste entre parenthèses de paires attributs/valeurs
body disposition	Liste entre parenthèses constituée d'une chaîne de type de disposition suivie d'une liste entre parenthèses de paires attributs/valeurs comme défini dans [DISPOSITION]
body language	Chaîne ou liste entre parenthèses indiquant le langage du corps comme défini dans [LANGUAGE-TAGS]

  Les données d'extension ultérieures ne sont pas encore définies dans cette version du protocole. Ces données d'extension peuvent être constituées ou non de NIL(s), de chaînes, de nombres ou de listes entre parenthèses de ces données. Les implémentations clientes utilisant l'élément de donnée BODYSTRUCTURE doivent prévoir le traitement de ces données d'extension. Les implémentations Serveur ne doivent pas envoyer de telles données d'extension tant qu'elles ne sont pas définies dans une révision du protocole.
  Exemple : *BODYSTRUCTURE (("TEXT" "PLAIN" ("CHARSET" "us-ascii") NIL NIL "7BIT" 22 1 NIL NIL NIL)("APPLICATION" "BYTE-STREAM" ("NAME" "casta37.jpg" "X-MAC-TYPE" "4A504547" "X-MAC-CREATOR" "6F676C65") NIL NIL "BASE64" 98642 NIL ("ATTACHMENT" ("FILENAME" "casta37.jpg")) NIL) "MIXED" ("BOUNDARY" "4D_====================1385356==") NIL NIL)*
  Description : *(("bodytype" "bodysubtype" (BodyParameterParenthesizedList) bodyId bodyDescription "bodyEncoding" BodySize BodySizeInTextLines ExtensionDataBODYmd5 ExtensionDataBodyDisposition ExtensionDataBodyLanguage) ("bodytype" "bodysubtype" (BodyParameterParenthesizedList) bodyId bodyDescription "bodyEncoding" BodySize BodySizeInTextLines ExtensionDataBODYmd5 ExtensionDataBodyDisposition ExtensionDataBodyLanguage) "multipartSubtype" (ExtensionDataBodyParameterList) ExtensionDataBodyDisposition ExtensionDataBodyLanguage))*

- **UID**
  Récupère le numéro d'identification unique du message — équivaut à l'exécution de *IMAP_UIDToMsgNum*. Comme ce numéro est retourné dans une zone de texte, vous devrez le convertir en Entier long.
  Exemple :

```
données Msg:="UID"
$Err:=IMAP_MsgFetch(imap_ID;1;données Msg;valeursMsg)
```

valeursMsg retourne UID 250000186
Pour obtenir un entier long :

```
C_LONGINT(vLongint)
VLongint:=Num("250000186")
```

Macro éléments de données

- **FAST**
  Macro équivalente à : (FLAGS INTERNALDATE RFC822.SIZE)
  Exemple :

```
$Err:=IMAP_MsgFetch(imap_ID;msgNum;"FAST";valeursMsg)
```

valeursMsg retourne *"FLAGS (\Seen \Answered) INTERNALDATE “17-Jul-2001 15:45:37 +0200” RFC822.SIZE 99599"*

- **ALL**
  Macro équivalente à : (FLAGS INTERNALDATE RFC822.SIZE ENVELOPE)
- **FULL**
  Macro équivalente à : (FLAGS INTERNALDATE RFC822.SIZE ENVELOPE BODY)

IMAP_MsgInfo

IMAP_MsgInfo (imap_ID ; numéroMsg ; tailleMsg ; uniqueID) -> Résultat

Paramètre	Type		Description
imap_ID	Entier long	➡	Référence de connexion IMAP
numéroMsg	Entier long	➡	Numéro du message
tailleMsg	Entier long	⬅	Taille du message
uniqueID	Entier long	⬅	ID unique d'un message sur le serveur
Résultat	Entier	↩	Code d'erreur

Description

La commande *IMAP_MsgInfo* retourne la taille et le numéro d'ID unique du message désigné par *numéroMsg* dans la boîte aux lettres courante.

imap_ID contient la référence d'une session ouverte avec *IMAP_Login*.

Le paramètre *numéroMsg* indique le message sur lequel vous souhaitez obtenir des informations. Le *numéroMsg* représente la position du message dans la liste courante des messages. Attention, le *numéroMsg* d'un message n'est pas une valeur stable, il diffère d'une session à l'autre en fonction des opérations effectuées.

Le paramètre *tailleMsg* retourne la taille du message.

Le paramètre *uniqueID* retourne le numéro d'ID unique du message sur le serveur. L'*uniqueID* est une valeur affectée au message par le serveur IMAP4. Cette valeur ne varie pas d'une session à l'autre, à la différence de *numéroMsg*. La valeur *uniqueID* est une référence utile pour vérifier si la base de données a déjà téléchargé un message depuis le serveur.

IMAP_MsgLst

IMAP_MsgLst (imap_ID ; premierMsg ; dernierMsg ; tabEnTêtesMsg ; tabNumMsg ; tabIDMsg ; tabValeursMsg) -> Résultat

Paramètre	Type		Description
imap_ID	Entier long	→	Référence de connexion IMAP
premierMsg	Entier long	→	Numéro du premier message
dernierMsg	Entier long	→	Numéro du dernier message
tabEnTêtesMsg	Tableau chaîne	→	Tableau des en-têtes à récupérer
tabNumMsg	Tableau entier long	←	Tableau des numéros de messages
tabIDMsg	Tableau entier long	←	Tableau ID uniques
tabValeursMsg	Tableau alpha 2D, Tableau texte 2D	←	Tableau 2D des valeurs des en-têtes
Résultat	Entier	↩	Code d'erreur

Description

La commande *IMAP_MsgLst* permet d'obtenir des informations spécifiques sur le contenu d'une boîte aux lettres. Seules les valeurs des en-têtes peuvent être récupérées par cette commande. Le contenu des en-têtes est automatiquement décodé et converti si nécessaire (reportez-vous à la description de la commande *POP3_Charset* pour plus d'informations sur les règles de conversion et de décodage).

imap_ID contient la référence d'une session ouverte avec *IMAP_Login*.

Le paramètre *premierMsg* désigne le numéro du premier message à examiner. Ce numéro représente la position d'un message dans la liste de tous les messages de la boîte aux lettres courante.

Le paramètre *dernierMsg* indique le numéro du dernier message à examiner. Ce numéro représente la position d'un message dans la liste de tous les messages de la boîte aux lettres courante.

Note : Si le paramètre *premierMsg* est supérieur au paramètre *dernierMsg*, les commandes *IMAP_Delete*, *IMAP_MsgLstInfo*, *IMAP_MsgLst*, *IMAP_SetFlags*, *IMAP_GetFlags* et *IMAP_CopyToMB* ne retournent pas d'erreur et ne font rien.

Vous remplissez le tableau alphanumérique ou texte *tabEnTêtesMsg* avec les noms des en-têtes que vous souhaitez récupérer.

Le tableau *tabNumMsg* retourne les numéros des messages compris entre *premierMsg* et *dernierMsg*.

Le tableau *tabIDMsg* reçoit les ID uniques de chaque message.

Le tableau 2D *tabValeursMsg* reçoit le contenu de chaque en-tête désigné par *tabEnTêtesMsg*. A chaque en-tête demandé correspond une "ligne" du tableau *tabValeursMsg*.

Exemple

```
aEnTêtes{1}:="Date:"
aEnTêtes{2}:="From:"
aEnTêtes{3}:="Subject:"
IMAP_MsgLst(IMAP_ID;vPremier;vDernier;aEnTêtes;aNumMsg;aUID;aValeurs)
```

aValeurs{1}{1} contient par exemple "Jeudi 19 novembre 1998, 00:24:02 -0800"
aValeurs{2}{1} contient par exemple "Jack@4d.com"
aValeurs{3}{1} contient par exemple "Appelez votre femme"

Les erreurs sont gérées de la façon suivante :
1) Seules les erreurs relatives à la communication sont renvoyées. Si la commande ne peut pas achever sa tâche en raison d'une erreur (réseau, syntaxe, serveur, etc.), le code d'erreur approprié est renvoyé.
2) Si un message appartenant à l'intervalle spécifié n'existe pas ou comporte une erreur :
- Aucun élément de tableau n'est créé pour ce message.
- Aucune erreur n'est renvoyée.
3) L'incapacité à localiser un ou plusieurs en-têtes dans un message ne constitue pas une erreur :
- Un élément de tableau est créé pour le message.
- Les éléments des tableaux *aNumMsg* et *aUID* contiennent les valeurs appropriées.
- Pour chaque en-tête introuvable dans le message, une chaîne vide ("") est renvoyée à l'élément de tableau.
- Aucun code d'erreur n'est renvoyé.

IMAP_MsgLstInfo

IMAP_MsgLstInfo (imap_ID ; premierMsg ; dernierMsg ; tabTaillesMsg ; tabNumMsg ; tabIDMsg) -> Résultat

Paramètre	Type		Description
imap_ID	Entier long	→	Référence de connexion IMAP
premierMsg	Entier long	→	Numéro du premier message
dernierMsg	Entier long	→	Numéro du dernier message
tabTaillesMsg	Tableau entier long	←	Tableau des tailles
tabNumMsg	Tableau entier long	←	Tableau des numéros de messages
tabIDMsg	Tableau entier long	←	Tableau des ID uniques
Résultat	Entier	↩	Code d'erreur

Description

La commande *IMAP_MsgLstInfo* retourne la taille, le numéro et l'ID unique d'un ensemble de messages contenus dans la boîte aux lettres courante (définie par la commande *IMAP_SetCurrentMB*). Ces informations sont renvoyées dans trois tableaux, chaque élément de tableau correspondant à un message. Les tableaux passés en paramètres doivent être déclarés, en revanche leur nombre d'éléments n'a pas d'importance : la commande *IMAP_MsgLstInfo* les redimensionne au nombre de messages récupérés.

La commande *IMAP_MsgLstInfo* ne renvoie pas d'erreur si elle ne parvient pas à récupérer des informations sur un message particulier. Dans ce cas, la commande ignore le message et ne crée pas d'élément correspondant dans les tableaux. Par conséquent, si la commande réussit à lire tous les messages, le tableau *tabNumMsg* contiendra une suite continue de valeurs numériques séquentielles. En cas de problème, des numéros seront manquants dans la suite numérique de *tabNumMsg*.

imap_ID contient la référence d'une session ouverte avec *IMAP_Login*.

Le paramètre *premierMsg* désigne le numéro du premier message à examiner. Ce numéro représente la position d'un message dans la liste de tous les messages de la boîte aux lettres courante.

Le paramètre *dernierMsg* indique le numéro du dernier message à examiner. Ce numéro représente la position d'un message dans la liste de tous les messages de la boîte aux lettres courante.

Note : Si le paramètre *premierMsg* est supérieur au paramètre *dernierMsg*, les commandes *IMAP_Delete*, *IMAP_MsgLstInfo*, *IMAP_MsgLst*, *IMAP_SetFlags*, *IMAP_GetFlags* et *IMAP_CopyToMB* ne retournent pas d'erreur et ne font rien.

Le tableau *tabTaillesMsg* reçoit la taille de chaque message dont le numéro est compris entre *premierMsg* et *dernierMsg*.

Le tableau *tabNumMsg* reçoit les numéros de chaque message.

Le tableau *tabIDMsg* reçoit les ID uniques de chaque message.

IMAP_MsgNumToUID

IMAP_MsgNumToUID (imap_ID ; numéroMsg ; uniqueID) -> Résultat

Paramètre	Type		Description
imap_ID	Entier long	➡	Référence de connexion IMAP
numéroMsg	Entier long	➡	Numéro du message
uniqueID	Entier long	⬅	ID unique d'un message sur le serveur
Résultat	Entier	↩	Code d'erreur

Description

La commande *IMAP_MsgNumToUID* récupère le numéro d'ID unique d'un message de la boîte aux lettres courante référencée par *imap_ID* à partir de son numéro courant dans la liste des messages.

imap_ID contient la référence d'une session ouverte avec *IMAP_Login*.

Le paramètre *numéroMsg* contient le numéro courant du message (la position du message dans la liste de messages courante).

Le paramètre *uniqueID* retourne le numéro d'identification unique du message sur le serveur IMAP.

Si le *numéroMsg* ne peut pas être trouvé sur le serveur, *uniqueID* retourne 0 (zéro), aucune erreur n'est renvoyée.

IMAP_RenameMB

IMAP_RenameMB (imap_ID ; nomBL ; nouvNomBL) -> Résultat

Paramètre	Type		Description
imap_ID	Entier long	➡	Référence de connexion IMAP
nomBL	Texte	➡	Nom de la boîte aux lettres à renommer
nouvNomBL	Texte	➡	Nouveau nom de boîte aux lettres
Résultat	Entier	⮌	Code d'erreur

Description

La commande *IMAP_RenameMB* renomme une boîte aux lettres. Une erreur est générée si vous tentez de renommer une boîte aux lettres qui n'existe pas ou si vous utilisez un nom de boîte aux lettres existante.

Note : Il est possible de renommer la boîte aux lettres INBOX. Dans ce cas, tous les messages de la boîte INBOX sont déplacés dans une nouvelle boîte portant le nouveau nom, laissant ainsi la boîte INBOX vide. Si le serveur IMAP autorise la création de sous-boîtes dans la boîte INBOX, elles ne sont pas affectées par le changement de nom.

imap_ID contient la référence d'une session ouverte avec *IMAP_Login*.

nomBL contient le nom complet de la boîte aux lettres à renommer.

nouvNomBL contient le nouveau nom complet de la boîte aux lettres.

IMAP_Search

IMAP_Search (imap_ID ; critèreRech ; tabNumMsg) -> Résultat

Paramètre	Type		Description
imap_ID	Entier long	➡	Référence de connexion IMAP
critèreRech	Texte	➡	Critère(s) de recherche
tabNumMsg	Tableau entier long	⬅	Tableau des numéros de messages
Résultat	Entier	↩	Code d'erreur

Description

La commande *IMAP_Search* recherche les messages correspondant aux critères définis dans la boîte aux lettres courante. Le paramètre *critèreRech* contient un ou plusieurs mots-clés de recherche. *tabNumMsg* retourne la liste des numéros des messages trouvés par la recherche.

imap_ID contient la référence d'une session ouverte avec *IMAP_Login*.

Le paramètre *critèreRech* contient un ou plusieurs mot(s)-clé(s) de recherche (cf. paragraphe "Mots-clés de recherche" ci-dessous) associé(s) ou non à des valeurs de recherche. Un critère de recherche peut être un mot-clé simple ou une liste de mots-clés entre parenthèses. Par exemple:

MotClé1 = FLAGGED
MotClé2 = NOT FLAGGED
MotClé3 = FLAGGED DRAFT

Note : Généralement, la recherche ne tient pas compte de la casse des caractères.

- Si le paramètre *critèreRech* est une chaîne vide, la recherche équivaudra à un "tout sélectionner" :

```
IMAP_Search(imap_ID;"";tabNumMsg)
```

... retourne tous les messages de la boîte aux lettres courante.

- Si le paramètre *critèreRech* contient plusieurs mots-clés de recherche, ils seront reliés par l'opérateur **ET** (intersection). Seuls les messages correspondant au premier ET au deuxième ET au troisième critère (etc.) seront trouvés.
- *critèreRech = FLAGGED FROM "SMITH"*
  ... retourne tous les messages comportant le marqueur **\Flagged** ET envoyés par Smith.

- Vous pouvez également utiliser les opérateurs **OR** (OU) ou **NOT** (SAUF) :
  critèreRech = OR SEEN FLAGGED
  ... retourne tous les messages comportant le marqueur **\Seen** OU **\Flagged**.
  critèreRech = NOT SEEN
  ... retourne tous les messages ne comportant pas le marqueur **\Seen**.
  critèreRech = HEADER CONTENT-TYPE "MIXED" NOT HEADER CONTENT-TYPE "TEXT"
  ... retourne tous les messages dont l'en-tête content-type contient "Mixed" et ne contient pas "Text".
  critèreRech = HEADER CONTENT-TYPE "E" NOT SUBJECT "o" NOT HEADER CONTENT-TYPE "MIXED"
  ... retourne tous les messages dont l'en-tête content-type contient "e" et dont l'en-tête Subject ne contient pas "o" et dont l'en-tête content-type n'est pas "Mixed".
  critèreRech = OR (ANSWERED SMALLER 400) (HEADER CONTENT-TYPE "E" NOT SUBJECT "o" NOT HEADER CONTENT-TYPE "MIXED")
  ... retourne tous les messages correspondant à la première liste de mots-clés entre parenthèses OU à la seconde liste.
  critèreRech = OR ANSWERED SMALLER 400 (HEADER CONTENT-TYPE "E" NOT SUBJECT "o" NOT HEADER CONTENT-TYPE "MIXED")
  ... retourne tous les messages comportant le marqueur **\Answered** OU dont la taille est inférieure à 400 octets ET répondant aux critères définis dans la liste entre parenthèses.

Dans les deux derniers exemples, le résultat obtenu sera différent si vous enlevez les parenthèses de la première liste de mots-clés.

- Le paramètre *critèreRech* peut contenir optionnellement l'instruction *[CHARSET]*. Cette instruction est composée du mot "CHARSET" suivi d'un jeu de caractères défini (US ASCII, ISO-8859). Elle indique le codage de caractères utilisé dans la chaîne de recherche. Par conséquent, vous devez convertir la chaîne de recherche dans le jeu de caractères spécifié si vous utilisez l'instruction [CHARSET] (cf. la

commande de 4D **Mac vers ISO**).
Par défaut, 4D Internet Commands encode la chaîne de critères de recherche en "Quotable Printable" si elle contient des caractères étendus.
critèreRech = CHARSET "ISO-8859" BODY "Help"
... signifie que le critère de recherche utilise le jeu de caractères ISO-8859 et que le serveur devra convertir la chaîne avant de débuter la recherche, si nécessaire.

Types de valeurs de recherche

Les mots-clés de recherche peuvent traiter des valeurs des types suivants :

- **Valeurs de type date**

Les valeurs de type *<date>* sont placées dans des chaînes formatées de la manière suivante : *jour+"-"+mois+"-"+année* où *jour* indique la date du jour dans le mois (2 caractères maxi), *mois* indique le mois (Jan/Feb/Mar/Apr/May/Jun/Jul/Aug/Sep/Oct/Dec) et *année* indique l'année sur 4 chiffres.
Exemple : *critèreRech = SENTBEFORE 1-Feb-2000* (il n'est généralement pas nécessaire de mettre une date entre guillemets puisqu'elle ne contient pas de caractères spéciaux).

- **Valeurs de type chaîne**

Les valeurs de type *<chaîne>* peuvent contenir tout type de caractère et doivent être placées entre des guillemets. Toutefois, si la chaîne ne contient pas de caractères spéciaux (des espaces par exemple), les guillemets ne sont pas obligatoires. Dans tous les cas, les guillemets vous permettent de vous assurer que la chaîne sera correctement interprétée.
Exemple : *critèreRech = FROM "SMITH"*
Note : Les recherches basées sur des chaînes de caractères sont du type "contient" : si le champ d'un message contient au moins la chaîne recherchée, le message est trouvé. En outre, la recherche ne tient pas compte de la casse des caractères.

- **Noms de champs**

Les valeurs de type *<nom de champ>* contiennent le nom d'un champ d'en-tête.
Exemple : *critèreRech = HEADER CONTENT-TYPE "MIXED"*

- **Marqueurs**

Les valeurs de type *<marqueur>* acceptent un ou plusieurs mots-clés (y compris des marqueurs standard) séparés par des espaces.
Exemple : *critèreRech = KEYWORD \Flagged \Draft*

- **Ensemble de messages**

Les valeurs de ce type désignent un ensemble de messages. Elles contiennent une liste de numéros de messages dans un ordre croissant, de 1 au nombre total de messages dans la boîte aux lettres.
Les numéros sont séparés par des virgules ; un deux-points (:) indique un intervalle (inclusif) de numéros.
Exemples
*2,4:7,9,12:** représente les messages 2,4,5,6,7,9,12,13,14,15 pour une boîte contenant 15 messages.
critèreRech = 1:5 ANSWERED recherche, parmi les messages 1 à 5, ceux qui comportent le marqueur **\Answered**.
critèreRech = 2,4 ANSWERED recherche, parmi les messages 2 et 4, ceux qui comportent le marqueur **\Answered**.

Mots-clés de recherche

ALL
Tous les messages de la boîte aux lettres.
ANSWERED
Messages comportant le marqueur **\Answered**.
UNANSWERED
Messages ne comportant le marqueur **\Answered**.
DELETED
Messages comportant le marqueur **\Deleted**.
UNDELETED
Messages ne comportant pas le marqueur **\Deleted**.

DRAFT
Messages comportant le marqueur **\Draft**.
UNDRAFT
Messages ne comportant pas le marqueur **\Draft**.
FLAGGED
Messages comportant le marqueur **\Flagged**.
UNFLAGGED
Messages ne comportant pas le marqueur **\Flagged**.
RECENT
Messages comportant le marqueur **\Recent**.
OLD
Messages ne comportant pas le marqueur **\Recent**.
SEEN
Messages comportant le marqueur **\Seen**.
UNSEEN
Messages ne comportant pas le marqueur **\Seen**.
NEW
Messages comportant le marqueur **\Recent** et pas le marqueur **\Seen**. Equivaut à "(RECENT UNSEEN)".
KEYWORD <marqueur>
Messages comportant le marqueur spécifié.
UNKEYWORD <marqueur>
Messages ne comportant pas le marqueur spécifié.
BEFORE <date>
Messages dont la date interne est antérieure à la date spécifiée.
ON <date>
Messages dont la date interne est égale à la date spécifiée.
SINCE <date>
Messages dont la date interne est égale ou postérieure à la date spécifiée.
SENTBEFORE <date>
Messages dont l'en-tête Date est antérieur à la date spécifiée.
SENTON <date>
Messages dont l'en-tête Date est égal à la date spécifiée.
SENTSINCE <date>
Messages dont l'en-tête Date est égal ou postérieur à la date spécifiée.
TO <chaîne>
Messages contenant la chaîne spécifiée dans l'en-tête Destinataire.
FROM <chaîne>
Messages contenant la chaîne spécifiée dans l'en-tête Emetteur.
CC <chaîne>
Messages contenant la chaîne spécifiée dans l'en-tête CC.
BCC <chaîne>
Messages contenant la chaîne spécifiée dans l'en-tête BCC.
SUBJECT <chaîne>
Messages contenant la chaîne spécifiée dans l'en-tête Objet.
BODY <chaîne>
Messages dont le corps contient la chaîne spécifiée.
Texte <chaîne>
Messages contenant la chaîne spécifiée dans l'en-tête ou le corps.
HEADER <nom de champ> <chaîne>
Messages dont l'en-tête contient le champ défini et dont ce champ contient la chaîne définie.
UID <ID unique de message>
Messages dont le numéro unique correspond à la valeur spécifiée.
LARGER <valeur>
Messages dont la taille en octets est supérieure à la taille spécifiée.
SMALLER <valeur>

Messages dont la taille en octets est inférieure à la taille spécifiée.

NOT <critère>

Messages ne correspondant pas au critère spécifié.

OR <critère1> <critère2>

Messages correspondant au premier ou au deuxième critère spécifié.

IMAP_SetCurrentMB

IMAP_SetCurrentMB (imap_ID ; nomBL ; nbMsg ; nbNouvMsg ; marqListe ; marqPermanent ; uniqueIDBL) -> Résultat

Paramètre	Type		Description
imap_ID	Entier long	→	Référence de connexion IMAP
nomBL	Texte	→	Nom de boîte aux lettres
nbMsg	Entier long	←	Nombre de messages dans la boîte aux lettres spécifiée
nbNouvMsg	Entier long	←	Nombre de messages avec le marqueur \Recent
marqListe	Texte	←	Liste des marqueurs utilisés dans la boîte aux lettres
marqPermanent	Texte	←	Liste des marqueurs modifiables
uniqueIDBL	Entier long	←	Numéro d'identification unique de la boîte aux lettres spécifiée
Résultat	Entier	↩	Code d'erreur

Description

La commande *IMAP_SetCurrentMB* vous permet d'ouvrir une session (c'est-à-dire, de sélectionner une boîte aux lettres courante) et donc de gérer les messages de la boîte aux lettres spécifiée.

Une seule session peut être ouverte à la fois pendant une connexion ; les accès simultanés à plusieurs boîtes aux lettres nécessitent plusieurs connexions (plusieurs *IMAP_Login*). La commande *IMAP_SetCurrentMB* referme automatiquement la session courante avant de démarrer la nouvelle. Par conséquent, si une boîte aux lettres était définie comme courante et que l'exécution de la commande *IMAP_SetCurrentMB* échoue, vous n'avez plus de boîte aux lettres courante.

Vous pouvez fermer une session (c'est-à-dire, fermer la boîte aux lettres courante) sans sélectionner de nouvelle boîte aux lettres : il suffit d'exécuter la commande *IMAP_SetCurrentMB* avec un nom de boîte aux lettres (*nomBL*) inexistant puis la commande *IMAP_CloseCurrentMB* ou *IMAP_Logout* lors du traitement de l'erreur.

imap_ID contient la référence d'une session ouverte avec *IMAP_Login*.

Le paramètre *nomBL* contient le nom complet de la boîte aux lettres à rendre courante.

Le paramètre *nbMsg* retourne le nombre de messages présents dans la boîte aux lettres (retourne -1 en cas d'erreur).

Le paramètre *nbNouvMsg* retourne le nombre de messages récents présents dans la boîte aux lettres (retourne -1 en cas d'erreur).

Le paramètre *marqListe* retourne la liste des marqueurs utilisés dans la boîte aux lettres courante. Notez que seuls les marqueurs listés dans le paramètre *marqPermanent* peuvent être modifiés.

Le paramètre *marqPermanent* retourne la liste des marqueurs pouvant être modifiés de manière permanente (à l'exception du marqueur **\Recent**, géré par le serveur IMAP). Notez que la chaîne contenue dans le paramètre peut également inclure le marqueur spécial *, ce qui signifie que des mots-clés peuvent être créés en stockant ces marqueurs dans la boîte aux lettres (cf. commande *IMAP_SetFlags*). *marqPermanent* retourne une chaîne vide lorsque tous les marqueurs listés dans le paramètre *marqListe* peuvent être modifiés.

Le paramètre *uniqueIDBL* retourne le numéro d'identification unique de la boîte aux lettres courante. Cet identifiant est particulièrement utile lorsqu'une boîte aux lettres est supprimée puis qu'une nouvelle boîte est créée avec le même nom par la suite. Dans ce cas, seul le numéro unique permet au client d'identifier la nouvelle boîte aux lettres.

IMAP_SetFlags

IMAP_SetFlags (imap_ID ; premierMsg ; dernierMsg ; listeMarqMsg ; supprimerOption) -> Résultat

Paramètre	Type		Description
imap_ID	Entier long	➡	Référence de connexion IMAP
premierMsg	Entier long	➡	Numéro du premier message
dernierMsg	Entier long	➡	Numéro du dernier message
listeMarqMsg	Chaîne	➡	Marqueurs à ajouter ou supprimer
supprimerOption	Entier	➡	0 = ajouter marqueur, 1 = supprimer marqueur
Résultat	Entier	↩	Code d'erreur

Description

La commande *IMAP_SetFlags* permet d'ajouter ou de supprimer en une opération plusieurs marqueurs attachés aux messages de l'intervalle défini.

Le protocole IMAP permet d'associer une liste de marqueurs à un message. Il existe deux types de marqueurs : les **marqueurs permanents** et les **marqueurs de session**.
Les marqueurs permanents sont ajoutés ou supprimés de manière durable parmi les marqueurs des messages (cf. commande *IMAP_SetCurrentMB*) ; autrement dit, toute modification d'un marqueur permanent sera conservée lors des sessions ultérieures.
Les modifications effectuées sur les marqueurs de session ne sont valides que pendant la session.

Les marqueurs système actuellement définis sont les suivants :

- **Seen** : le message a été lu.
- **Answered** : une réponse au message a été envoyée.
- **Flagged** : le message est étiqueté "urgent" afin d'attirer l'attention.
- **Deleted** : le message est déclaré "à supprimer" et sera supprimé ultérieurement lors de l'exécution de la commande *IMAP_Delete*, *IMAP_CloseCurrentMB*, *IMAP_SetCurrentMB* ou *IMAP_Logout*.
- **Draft** : le message est encore à l'état de brouillon.
- **Recent** : le message a été reçu "récemment" dans la boîte aux lettres. Ce marqueur n'apparaît qu'au cours d'une seule session ; les sessions suivantes ne le verront pas. Ce marqueur permanent est géré par le serveur IMAP et ne peut pas être modifié par un client IMAP à l'aide de la commande *IMAP_SetFlags*, par exemple.

Certains serveurs IMAP permettent aux clients de définir des marqueurs personnalisés. Dans ce cas, les marqueurs additionnels sont appelés des mots-clés (keywords) et ne débutent pas par le caractère "\" (cf. commande *IMAP_SetCurrentMB*).

Note : Si vous positionnez le marqueur **\Deleted** puis refermez la session en exécutant la commande *IMAP_SetCurrentMB*, *IMAP_CloseCurrentMB*, *IMAP_Delete* ou *IMAP_Logout*, le message concerné sera supprimé définitivement.

imap_ID contient la référence d'une session ouverte avec *IMAP_Login*.

Le paramètre *premierMsg* désigne le numéro du premier message à examiner. Ce numéro représente la position d'un message dans la liste de tous les messages de la boîte aux lettres courante.

Le paramètre *dernierMsg* indique le numéro du dernier message à examiner. Ce numéro représente la position d'un message dans la liste de tous les messages de la boîte aux lettres courante.

Note : Si le paramètre *premierMsg* est supérieur au paramètre *dernierMsg*, les commandes *IMAP_Delete*, *IMAP_MsgLstInfo*, *IMAP_MsgLst*, *IMAP_SetFlags*, *IMAP_GetFlags* et *IMAP_CopyToMB* ne retournent pas d'erreur et ne font rien.

Vous pouvez passer un ou plusieurs marqueur(s) dans le paramètre *listeMarqMsg*. Si vous passez plusieurs marqueurs, ils doivent être séparés par des espaces (voir exemples ci-dessous).

Seuls les marqueurs permanents indiqués dans la liste *marqPermanent* pourront être modifiés (cf. commande *IMAP_SetCurrentMB*).

Le paramètre *optionSuppr* vous permet d'indiquer si vous souhaitez ajouter ou enlever le ou les marqueur(s) défini(s) dans le paramètre *listeMarqMsg* :

- 0 = ajout du ou des marqueur(s)

- 1 = suppression du ou des marqueur(s)

Exemple 1

Définition des marqueurs **\Answered** et **\Draft** pour les messages compris entre **StartMsg** et **EndMsg** :

```
msgFlagsName:="\Answered \Draft"
  ` \Answered et \Draft sont séparés par un espace
IMAP_SetFlags(imap_ID;startMsg;endMsg;msgFlagsName;0)
```

Exemple 2

Suppression du marqueur **\Deleted** pour les messages compris entre **StartMsg** et **EndMsg**, quel que soit l'état précédent de ce marqueur :

```
msgFlagsName:="\Deleted"
IMAP_SetFlags(imap_ID;startMsg;endMsg;msgFlagsName;1)
```

Exemple 3

Définition du marqueur **\Deleted** pour les messages compris entre **StartMsg** et **EndMsg**, quel que soit l'état précédent de ce marqueur :

```
msgFlagsName:="\Deleted"
IMAP_SetFlags(imap_ID;startMsg;endMsg;msgFlagsName;0)
IMAP_CloseCurrentMB(imap_ID)
  `Fermeture de la boîte aux lettres courante et suppression définitive des messages désignés
```

Exemple 4

Définition du marqueur **\Answered** en fonction de la valeur de la case à cocher "CheckBoxAnswered" :

```
$Error:=IMAP_SetFlags(vImap_ID;$msgNum;$msgNum;"\Answered";Num(CheckBoxAnswered=0))
```

IMAP_SetPrefs

IMAP_SetPrefs (retoursLigne ; dossierMsg) -> Résultat

Paramètre	Type		Description
retoursLigne	Entier	→	0 = Ne pas retirer les retours à la ligne, 1 = Retirer les retours à la ligne, -1 = Aucune modification
dossierMsg	Texte	→	Chemin d'accès au dossier des messages ("" = aucune modification)
Résultat	Entier	↩	Code d'erreur

Description

La commande *IMAP_SetPrefs* définit des préférences générales pour toutes les commandes IMAP ultérieures.

Le paramètre *retoursLigne* vous permet de préciser comment traiter les caractères de retour à la ligne dans les messages enregistrés. La plupart des serveurs IMAP associent un caractère Retour chariot (Carriage return) et un caractère Retour à la ligne (Line feed) pour indiquer la fin d'une ligne, à la différence des applications Macintosh qui requièrent un simple Retour chariot. Dans ce cas, cette option vous permet de supprimer les caractères Retour à la ligne superflus du texte des messages.

Le paramètre *dossierMsg* indique le chemin d'accès local du dossier dans lequel les messages récupérés à l'aide de la commande *IMAP_Download* doivent être enregistrés par défaut.

IMAP_SubscribeMB

IMAP_SubscribeMB (imap_ID ; nomBL ; abonnementBL) -> Résultat

Paramètre	Type		Description
imap_ID	Entier long	→	Référence de connexion IMAP
nomBL	Texte	→	Nom de la boîte aux lettres à laquelle s'abonner ou se désabonner
abonnementBL	Entier	→	0 = Se désabonner, 1 = S'abonner
Résultat	Entier	←	Code d'erreur

Description

La commande *IMAP_SubscribeMB* permet de modifier la liste des boîtes aux lettres du serveur auxquelles l'utilisateur est "abonné". Cette commande permet d'activer ou de supprimer l'abonnement à la boîte aux lettres définie par *nomBL*.

L'abonnement permet à l'utilisateur de restreindre la liste des boîtes aux lettres disponibles à celles qu'il consulte généralement. Pour cela, il suffit d'exécuter la commande *IMAP_ListMBs* en passant 1 dans le paramètre *abonnementsBL* (cf. commande *IMAP_ListMBs*).

imap_ID contient la référence d'une session ouverte avec *IMAP_Login*.

Le paramètre *nomBL* contient le nom complet de la boîte aux lettres à laquelle l'utilisateur souhaite s'abonner ou de laquelle il souhaite se désabonner.

Passez 0 dans le paramètre *abonnementBL* pour provoquer le désabonnement de l'utilisateur, et 1 pour l'abonner.

IMAP_UIDToMsgNum

IMAP_UIDToMsgNum (imap_ID ; uniqueID ; numéroMsg) -> Résultat

Paramètre	Type		Description
imap_ID	Entier long	➡	Référence de connexion IMAP
uniqueID	Entier long	➡	ID unique d'un message sur le serveur
numéroMsg	Entier long	⬅	Numéro du message
Résultat	Entier	↩	Code d'erreur

Description

La commande *IMAP_UIDToMsgNum* retourne le numéro de message *numéroMsg* correspondant au message désigné par *uniqueID* dans la boîte aux lettres référencée par *imap_ID*. Le *numéroMsg* d'un message étant une valeur fluctuante, relative aux autres messages de la liste, cette commande renvoie la position courante d'un message dont les informations peuvent avoir été récupérées à l'occasion d'une précédente session IMAP.

imap_ID contient la référence d'une session ouverte avec *IMAP_Login*.

Le paramètre *uniqueID* contient le numéro d'identification unique du message à localiser sur le serveur IMAP.

Le paramètre *numéroMsg* retourne le numéro courant du message (la position du message dans la liste de messages courante) désigné par *uniqueID*. Si l'*uniqueID* ne peut pas être trouvé sur le serveur, *numéroMsg* retourne 0 (zéro), aucune erreur n'est renvoyée.

IMAP_VerifyID

IMAP_VerifyID (imap_ID) -> Résultat

Paramètre	Type		Description
imap_ID	Entier long	➡	Référence de connexion IMAP
		⬅	0 = connexion déjà close
Résultat	Entier	↩	Code d'erreur

Description

Un serveur IMAP déconnecte automatiquement les comptes demeurés inactifs pendant une certaine période, déterminée par l'administrateur (ce mécanisme est appelé timeout). Chaque commande en interaction avec le serveur IMAP provoque la remise à zéro du compteur d'inactivité. La commande *IMAP_VerifyID* provoque également la remise à zéro du compteur d'inactivité de la connexion IMAP spécifiée, sans effectuer aucune autre action. Elle permet de conserver une session active en cas de risque de dépassement du délai.

Au moment de son exécution, la commande *IMAP_VerifyID* vérifie que la connexion n'a pas déjà été close. Si la connexion est toujours ouverte, la commande indique au serveur IMAP de remettre à zéro le compteur d'inactivité pour cette connexion. Si la connexion a déjà été fermée *IMAP_VerifyID* retourne l'erreur appropriée, libère la mémoire utilisée par la connexion IMAP et renvoie 0 dans le paramètre *imap_ID*.

Le paramètre *imap_ID* contient une référence de connexion ouverte par *IMAP_Login*.

IC Internet

- Commandes spéciales Internet, Présentation
- NET_AddrToName
- NET_Finger
- NET_NameToAddr
- NET_Ping
- NET_Resolve
- NET_Time

Commandes spéciales Internet, Présentation

Les commandes de ce thème permettent d'effectuer des tâches courantes sur Internet. A l'aide de ces commandes, vous pourrez ainsi sonder (Ping) et explorer (Finger) une machine, obtenir l'heure d'une horloge réseau, décoder un nom de domaine ou une adresse IP, ou encore les convertir en valeurs de type entier long. Ces commandes s'utilisent généralement en association avec les autres commandes Internet de 4D.

NET_AddrToName

NET_AddrToName (ip_EntierLong ; nomServeur ; adresse_IP) -> Résultat

Paramètre	Type		Description
ip_EntierLong	Entier long	⇒	Référence d'adresse
nomServeur	Chaîne	⇐	Nom du serveur
adresse_IP	Chaîne	⇐	Adresse IP
Résultat	Entier	↩	Code d'erreur

Description

La commande *NET_AddrToName* retourne le nom et l'adresse IP d'un serveur à partir de sa référence unique *ip_EntierLong*, générée par la commande *NET_NameToAddr*.

Important : Afin d'assurer la compatibilité future de vos applications avec IPV6, l'utilisation de cette commande est déconseillée. Il est préférable d'utiliser *NET_Resolve* qui travaille sur la base de chaînes de caractères.

ip_EntierLong indique la référence unique d'une adresse IP.

nomServeur retourne le nom du serveur sous forme de chaîne de caractères.

adresse_IP retourne l'adresse IP du serveur sous forme de chaîne de caractères.

Si le nom du serveur ne peut être converti, *adresse_IP* retourne une chaîne vide, aucune erreur n'est renvoyée.

NET_Finger

NET_Finger (nomServeur ; texteRecherche ; informations) -> Résultat

Paramètre	Type		Description
nomServeur	Chaîne	➡	Nom ou adresse IP du serveur
texteRecherche	Chaîne	➡	Texte de la recherche
informations	Texte	⬅	Résultats de la recherche
Résultat	Entier	↩	Code d'erreur

Description

La commande *NET_Finger* permet d'obtenir des informations sur un compte utilisateur enregistré sur un serveur. La commande Unix Finger retourne l'heure de la dernière connexion d'un utilisateur ainsi que des informations supplémentaires que l'utilisateur choisit de fournir dans ses fichiers ".plan" et ".project". *NET_Finger* interroge, sur la machine désignée par *nomServeur*, le compte d'utilisateur spécifié par *texteRecherche* et retourne le résultat dans le paramètre *informations*.

Une recherche Finger peut être effectuée de deux manières :

- **Directement :** la recherche est effectuée directement sur la machine de l'utilisateur. Par exemple, pour obtenir des informations sur "johnt" chez "4d.com", vous pouvez écrire :

```
$erreur:=NET_Finger("www.4d.com";"johnt";$textefinger)
```

- **Indirectement :** dans ce cas, vous demandez à un serveur distant (qui accepte la commande Finger) d'effectuer la requête. Par exemple, la requête suivante demande à la machine identifiée par le nom de domaine "4d.com" d'effectuer une recherche Finger de l'utilisateur "johnt@4d.com".

```
$erreur:=NET_Finger("www.4d.com";"johnt@4d.com";$textefinger)
```

Bien que les principales informations renvoyées dans chaque cas soient globalement identiques, de légères différences peuvent être constatées. En effet, les paramétrages d'exécution de la commande Finger peuvent différer d'une machine à l'autre. En outre, des différences de formatage peuvent apparaître entre les résultats d'une commande Finger directe et ceux d'une commande indirecte, les recherches indirectes contenant souvent des retours à la ligne (line feed) supplémentaires.

Le paramètre *nomServeur* contient le nom ou l'adresse IP du serveur sur lequel l'utilisateur identifié par *texteRecherche* a un compte.

Le paramètre *texteRecherche* contient soit le texte à rechercher sur un serveur Finger, soit le nom ou l'adresse IP d'une machine. Si vous passez un nom d'utilisateur dans *texteRecherche*, la commande le recherchera dans le répertoire des noms d'utilisateurs du serveur. Si vous passez un nom de machine ou une adresse IP, la commande enverra la requête Finger à la machine spécifiée par l'intermédiaire du serveur Finger *nomServeur*.

Le paramètre *informations* retourne le résultat de la recherche.

NET_NameToAddr

NET_NameToAddr (nomServeur ; ip_EntierLong) -> Résultat

Paramètre	Type		Description
nomServeur	Chaîne	⇒	Nom ou adresse IP du serveur
ip_EntierLong	Entier long	⇐	Référence à l'adresse
Résultat	Entier	⇔	Code d'erreur

Description

La commande *NET_NameToAddr* retourne un entier long qui référence de manière unique le nom ou l'adresse IP d'un serveur.

Important : Afin d'assurer la compatibilité future de vos applications avec IPV6, l'utilisation de cette commande est déconseillée. Il est préférable d'utiliser *NET_Resolve* qui travaille sur la base de chaînes de caractères.

Le paramètre *nomServeur* contient le nom ou l'adresse IP du serveur.

Le paramètre *ip_EntierLong* retourne une valeur identifiant l'adresse IP spécifiée dans le paramètre *nomServeur*. Toute chaîne d'adresse IP peut être convertie en entier long.

La commande *NET_NameToAddr* est utile pour économiser de la place lors du stockage des données. En effet, les valeurs stockées dans le format entier long sont plus compactes que les chaînes de caractères.

NET_Ping

NET_Ping (nomServeur ; texte ; actif ; timeOut) -> Résultat

Paramètre	Type		Description
nomServeur	Chaîne	⇒	Nom ou adresse IP du serveur
texte	Texte	⇒	Texte à envoyer dans le "ping"
actif	Entier	⇐	1 = Actif, 0 = Timeout/Inactif
timeOut	Entier	⇒	Nombre de secondes d'attente, 0 = utiliser la valeur IT_SetTimeOut
Résultat	Entier	↩	Code d'erreur

Description

La commande *NET_Ping* permet d'interroger une adresse IP distante pour vérifier qu'elle est active. Si la machine sondée exploite le protocole TCP/IP et si le réseau entre les deux sites est opérationnel, le statut 'Actif' est renvoyé.

Toute machine ayant une adresse IP accessible via le réseau mondial peut être sondée, y compris celles des utilisateurs finaux (toutefois, certains systèmes de sécurité appelés "firewalls" peuvent empêcher le sondage des machines placées sous leur protection).

Le paramètre *nomServeur* contient le nom de serveur ou l'adresse IP de la machine à sonder.

Le paramètre *texte* contient le texte à envoyer dans le "ping". Ce paramètre permet uniquement d'agir sur la taille du paquet TCP envoyé lors de l'exécution de la commande.

Le paramètre *actif* retourne un code indiquant l'état de la machine sondée. S'il reçoit la valeur 1, l'ordinateur est actif. S'il reçoit la valeur 0 (zéro), l'ordinateur est inactif ou la commande a dépassé le délai autorisé (timeout) sans recevoir de réponse.

Le paramètre *timeOut* spécifie le nombre maximum de secondes alloué pour l'exécution de la commande *NET_Ping*. Si ce paramètre est omis ou si vous passez 0, la valeur de timeout spécifiée par la commande *IT_SetTimeOut* sera utilisée.

Note : Le paramètre *timeOut* est ignoré sous Windows 95/98 et Millenium.

NET_Resolve

NET_Resolve (nomServeur ; ipOuServeur) -> Résultat

Paramètre	Type		Description
nomServeur	Chaîne	➡	Nom ou adresse IP du serveur
ipOuServeur	Chaîne	⬅	Renvoie la valeur opposée
Résultat	Entier	↩	Code d'erreur

Description

La commande *NET_Resolve* "décode" le nom ou l'adresse IP d'un serveur.

- Si vous passez un nom de serveur dans le paramètre *nomServeur*, la commande *NET_Resolve* renvoie l'adresse IP dans le paramètre *ipOuServeur*.
- Si vous passez une adresse IP dans le paramètre *nomServeur*, la commande *NET_Resolve* renvoie le nom de serveur enregistré pour cette machine dans le paramètre *ipOuServeur*.

Exemple

La méthode suivante fournit d'abord un nom de serveur "www.netcom.com" à la commande *NET_Resolve* afin d'obtenir son adresse IP. La méthode effectue ensuite un autre appel à la commande, en lui fournissant l'adresse IP afin d'obtenir son nom de serveur enregistré.

```
C_BOOLEAN($ERR)
C_TEXT($Converti)  //Peut être une chaîne de caractères ou un texte de toute taille
$ERR:=VérifErr("Convertir_NET";NET_Resolve("www.netcom.com";$Converti))
 //$Converti a renvoyé la valeur '192.100.81.100'
$ERR:=VérifErr("Convertir_NET";NET_Resolve($Converti;$Converti))
 //$Converti a renvoyé la valeur 'www.netcom.com'
```

Note : La méthode ***VérifErreur*** est détaillée dans la description de la commande **IT_ErrorText**.

NET_Time

NET_Time (nomServeur ; date ; délai ; décalage) -> Résultat

Paramètre	Type		Description
nomServeur	Chaîne	➡	Nom ou adresse IP du serveur NTP
date	Date	⬅	Date
délai	Entier long	⬅	Heure, exprimée en secondes depuis minuit
décalage	Entier	➡	Nombre d'heures de décalage
Résultat	Entier	↩	Code d'erreur

Description

La commande *NET_Time* permet de récupérer la date et l'heure courantes d'une horloge réseau sur Internet, et de leur appliquer le décalage nécessaire pour la conversion en heure locale de l'utilisateur.

Note : Cette commande n'affecte pas l'horloge interne de l'ordinateur.

Le paramètre *nomServeur* contient le nom ou l'adresse IP d'un serveur NTP (Network Time Protocol).

Le paramètre *laDate* retourne la date (au format date 4D) fournie par le serveur NTP et à laquelle le *décalage* a été appliqué.

Le paramètre *heure* retourne l'heure fournie par le serveur NTP, après l'application du *décalage*. Cette valeur représente le nombre de secondes écoulées depuis minuit à cette *date*. L'exemple suivant propose une méthode de conversion de cette valeur en une variable heure 4D.

Le paramètre *décalage* indique le nombre d'heures à ajouter ou à soustraire des valeurs reçues. Les horloges réseau d'Internet expriment leurs valeurs en temps universel (TU). Même si l'horloge réseau située est dans votre fuseau horaire, il est probable que vous deviez fournir une valeur de *décalage* pour compenser la différence entre votre heure locale et le temps universel.

Exemple

L'exemple suivant récupère la date et l'heure de l'horloge réseau située sur le site "apple.com". La commande soustrait ensuite les sept heures de décalage spécifiées et renvoie la date et l'heure résultantes (l'heure est exprimée sous forme d'un entier long, qui peut ensuite être converti à l'aide de la commande 4D **Chaine heure**, comme illustré ci-dessous).

```
C_DATE(vDateNet)
C_LONGINT(vHeureNet)
C_TIME(vHeure)
C_LONGINT(vDécalage)
If(VérifErreur("Heure_Net";NET_Time("www.apple.com";vDateNet;vHeureNet;-7)))
   vHeure:=Time(Time string(vHeureNet))  //Convertit l'heure entier long en heure 4D
End if
```

Note : La méthode ***VérifErreur*** est détaillée dans la description de la commande *IT_ErrorText*.

IC POP3 Review Mail

- Réception de courrier, Présentation
- POP3_BoxInfo
- POP3_Charset
- POP3_Delete
- POP3_DownLoad
- POP3_GetMessage
- POP3_GetPrefs
- POP3_Login
- POP3_Logout
- POP3_MsgInfo
- POP3_MsgLst
- POP3_MsgLstInfo
- POP3_Reset
- POP3_SetPrefs
- POP3_UIDToNum
- POP3_VerifyID

Réception de courrier, Présentation

Les commandes POP3 permettent à votre base de données de récupérer des messages d'un serveur de courrier POP3. Les commandes Internet de 4D sont conformes aux spécifications MIME, elles peuvent reconnaître et extraire des messages contenant plusieurs pièces jointes.

Les commandes POP3 sont réparties en deux thèmes, "IC POP3 Review Mail" et "IC Downloaded Mail", correspondant aux deux modes de lecture du courrier électronique. Le premier mode consiste à prendre connaissance du contenu du courrier, à le télécharger. Le second mode consiste à travailler sur les messages téléchargés.

La taille des fichiers à télécharger va déterminer l'utilisation d'un mode par rapport à l'autre. Par exemple, un seul message électronique auquel est joint un fichier de 5 Mo pourrait facilement dépasser la capacité de stockage de la base de données. Seul un BLOB ou une image 4D est capable d'accueillir des données de cette taille, mais la conversion d'un message ou d'un document joint dans ce format est souvent inefficace car la messagerie cliente doit mobiliser de grandes ressources mémoire pour accéder à l'image ou au BLOB. Pour résoudre ce problème, la commande *POP3_Download* transfère un message du serveur POP3 directement sur le disque local de l'utilisateur. Il suffit ensuite d'utiliser les commandes du thème "IC Downloaded Mail" pour manipuler le fichier sur disque.

L'utilisation des commandes POP3 nécessite une bonne compréhension des paramètres *numéroMsg* et *uniqueID*. *numéroMsg* représente le numéro d'un message dans la boîte aux lettres au moment de l'exécution de la commande *POP3_Login*. A la connexion, les messages de la boîte aux lettres sont numérotés de 1 à x (x étant le nombre d'éléments présents dans la boîte aux lettres). Les numéros sont affectés en fonction de l'ordre dans lequel les messages ont été reçus, le numéro 1 étant le plus ancien. Les numéros affectés aux messages ne sont valides que pendant la période comprise entre *POP3_Login* et *POP3_Logout*.

Au moment de l'exécution de *POP3_Logout*, tout message marqué comme "devant être supprimé" disparaît. Lorsque l'utilisateur se reconnecte au serveur, les messages présents dans la boîte aux lettres sont de nouveau numérotés de 1 à x. Par exemple, s'il y a 10 messages dans la boîte aux lettres, et si les messages numérotés de 1 à 5 sont supprimés, les messages 6 à 10 seront renumérotés de 1 à 5 la prochaine fois que l'utilisateur consultera sa boîte aux lettres.

Pour illustrer ce fonctionnement, supposons que vous vous connectiez à un serveur POP3 et obteniez la liste de messages suivante :

numéroMsg	uniqueID	Date	De	Objet
1	bd573a4dbd573a4d	1 Jul 1998 …	jimw@acme.com	Clients potentiels …
2	bd574dc7bd574dc7	1 Jul 1998 …	frank@acme.com	Commande de licence sur site
3	bd575f06bd575f06	3 Jul 1998 …	joe@acme.com	Qui veut déjeuner ?
4	bd5761d4bd5761d4	4 Jul 1998 …	kelly@acme.com	Appel de votre femme…
5	bd577dc7db577dc5	4 Jul 1995 …	track@fedex.com	Suivi FedEx

Pendant la session, vous supprimez les messages 3 et 4. Lorsque vous quittez la session, vos demandes de suppression sont exécutées. Lorsque vous retournez sur le serveur, la liste de messages est alors renumérotée ainsi :

numéroMsg	uniqueID	Date	De	Objet
1	bd573a4dbd573a4d	1 Jul 1998 …	jimw@acme.com	Clients potentiels …
2	bd574dc7bd574dc7	1 Jul 1998 …	frank@acme.com	Commande de licence sur site
3	bd577dc7db577dc5	4 Jul 1995 …	track@fedex.com	Suivi FedEx

numéroMsg n'est pas une valeur statique se rapportant à un message spécifique, elle indique la position relative d'un message de la boîte aux lettres au moment de l'ouverture de la session.

En revanche, *uniqueID* est un numéro unique affecté au message lors de sa réception par le serveur. Ce numéro est défini par le serveur POP3 sur la base de l'heure et de la date auxquelles le message est reçu. Malheureusement, les serveurs POP3 n'utilisent pas *uniqueID* comme référence principale des messages. Aussi, lorsque vous manipulez des messages avec les commandes POP3, vous devez passer *numéroMsg* comme paramètre d'identification des messages sur le serveur. Par conséquent, vous devez être prudent lorsque vous développez des applications qui référencent des messages dans la base de données tout en laissant le contenu du message sur le serveur.

Note : Pour une plus grande souplesse, les commandes Internet de 4D permettent de passer directement une référence de connexion POP3 aux commandes TCP de bas niveau et inversement. Pour plus d'informations, reportez-vous à la section **Routines de bas niveau, Présentation**.

POP3_BoxInfo

POP3_BoxInfo (pop3_ID ; nombreMsg ; tailleMsg) -> Résultat

Paramètre	Type		Description
pop3_ID	Entier long	➡	Référence d'une connexion POP3
nombreMsg	Entier long	⬅	Nombre de messages
tailleMsg	Entier long	⬅	Taille des messages
Résultat	Entier	↩	Code d'erreur

Description

La commande *POP3_BoxInfo* renvoie le nombre et la taille des messages présents dans la boîte aux lettres de la session référencée par *pop3_ID*.

pop3_ID contient la référence d'une session ouverte avec *POP3_Login*.

nombreMsg retourne le nombre de messages présents dans la boîte aux lettres.

tailleMsg retourne la taille totale des messages présents dans la boîte aux lettres.

POP3_Charset

POP3_Charset (décoderEntêtes ; jeuCorps) -> Résultat

Paramètre	Type		Description
décoderEntêtes	Entier	➡	-1 = Utiliser le paramétrage courant, 0 = Ne rien faire, 1 = Convertir dans le jeu de caractères Mac OS si ISO-8859-1 ou ISO-2022-JP, décoder les caractères étendus
jeuCorps	Entier	➡	-1 = Utiliser le paramétrage courant, 0 = Ne rien faire, 1 = Convertir dans le jeu de caractères Mac OS si ISO-8859-1 ou ISO-2022-JP
Résultat	Entier	⮌	Code d'erreur

Description

La commande *POP3_Charset* automatise le traitement des caractères étendus dans les messages lors de leur exploitation via certaines commandes POP3 et MSG. Si cette commande n'est pas appelée ou si ses deux paramètres sont mis à 0, les commandes Internet de 4D version 6.7 ou supérieure fonctionneront de la même manière qu'en version 6.5.x.

La commande *POP3_Charset* permet de définir, d'une part, si les en-têtes comportant des caractères étendus doivent être décodés et, d'autre part, si le jeu de caractères utilisé dans le corps des messages et dans les en-têtes doit être converti.

Cette commande est particulièrement utile pour le traitement des caractères étendus dans les en-têtes tels que "Subject" et les noms placés dans les adresses (par exemple, pour le décodage d'adresses sous la forme *=?ISO-8859-1?Q?Test=E9?= <test@n.net>*).

Le paramètre *decoderEntêtes* définit les traitements à appliquer aux champs d'en-tête lors de l'exécution des commandes *POP3_MsgLst* et *MSG_FindHeader* (voir Note de compatibilité). Par défaut, ce paramètre a pour valeur 0.

- -1 : Utiliser les paramétrages courants ;
- 0 : Ne rien faire ;
- 1 : L'en-tête est décodé si nécessaire. Si l'en-tête est décodé et si le jeu de caractères spécifié est de l'ISO-8859-1 ou de l'ISO-2022-JP, il est converti, respectivement en ASCII Mac OS ou en Shift-JIS.

Note de compatibilité (version 6.8.1) : *POP3_Charset* s'applique à la commande *MSG_FindHeader* uniquement si la commande *MSG_Charset* n'a pas été préalablement exécutée.

Le paramètre *jeuCorps* définit les traitements à appliquer au corps du message lors de l'exécution de la commande *MSG_GetBody* (voir Note de compatibilité). Par défaut, ce paramètre a pour valeur 0.

- -1 : Utiliser les paramètrages courants ;
- 0 : Ne rien faire ;
- 1 : Si le jeu de caractères spécifié dans le champ "Body-Content-Type" est de l' ISO-8859-1 ou de l'ISO-2022-JP, le texte du corps du message est converti, respectivement en ASCII Mac OS ou en Shift-JIS.

Note de compatibilité (version 6.8.1) : *POP3_Charset* s'applique à la commande *MSG_GetBody* uniquement si la commande *MSG_Charset* n'a pas été préalablement exécutée.

Exemple 1

Avec les commandes Internet de 4D version 6.5.x :

```
$Err:=MSG_FindHeader($fichMsg;"From";$from)
$from:=ISO to Mac($from)
$Err:=MSG_FindHeader($fichMsg;"To";$to)
$to:=ISO to Mac($to)
$Err:=MSG_FindHeader($fichMsg;"Cc";$cc)
$cc:=ISO to Mac($cc)
$Err:=MSG_FindHeader($fichMsg;"Subject";$subject)
$subject:=ISO to Mac($subject)

$Err:=MSG_MessageSize($fichMsg;$tailleEntete;$tailleCorps;$tailleMsg)
```

```
$Err:=MSG_GetBody($fichMsg;0;$tailleCorps;$Corps)
$Corps:=ISO to Mac($Corps)
```

Exemple 2

Avec les commandes Internet de 4D version 6.7.x :

```
$Err:=POP3_Charset(1;1)
$Err:=MSG_FindHeader($fichMsg;"From";$from)
$Err:=MSG_FindHeader($fichMsg;"To";$to)
$Err:=MSG_FindHeader($fichMsg;"Cc";$cc)
$Err:=MSG_FindHeader($fichMsg;"Subject";$subject)

$Err:=MSG_MessageSize($fichMess;$tailleEntete;$tailleCorps;$tailleMsg)
$Err:=MSG_GetBody($fichMess;0;$tailleCorps;$Corps)
```

Exemple 3

Avec les commandes Internet de 4D version 6.8.x :

```
$Err:=MSG_Charset(1;1)
$Err:=MSG_FindHeader($fichMsg;"From";$from)
$Err:=MSG_FindHeader($fichMsg;"To";$to)
$Err:=MSG_FindHeader($fichMsg;"Cc";$cc)
$Err:=MSG_FindHeader($fichMsg;"Subject";$subject)

$Err:=MSG_MessageSize($fichMess;$tailleEntete;$tailleCorps;$tailleMsg)
$Err:=MSG_GetBody($fichMess;0;$tailleCorps;$Corps)
```

POP3_Delete

POP3_Delete (pop3_ID ; premierMsg ; dernierMsg) -> Résultat

Paramètre	Type		Description
pop3_ID	Entier long	➡	Référence à une connexion POP3
premierMsg	Entier long	➡	Numéro du premier message
dernierMsg	Entier long	➡	Numéro du dernier message
Résultat	Entier	⮌	Code d'erreur

Description

Dans l'intervalle de messages compris entre *premierMsg* et *dernierMsg*, la commande *POP3_Delete* marque chaque message comme étant "à supprimer". La suppression réelle des messages ne sera effective que lorsque la commande *POP3_Logout* sera correctement exécutée. Si la session courante s'interrompt pour une raison quelconque (délai dépassé, panne de réseau, etc.) avant l'appel à la commande *POP3_Logout*, les messages marqués ne seront pas supprimés sur le serveur POP3.

Le paramètre *pop3_ID* contient la référence d'une session ouverte avec *POP3_Login*.

Le paramètre *premierMsg* désigne le numéro du premier message à supprimer.

Le paramètre *dernierMsg* désigne le numéro du dernier message à supprimer.

Note : Si le paramètre *premierMsg* est supérieur au paramètre *dernierMsg*, cette commande ne fait rien. Aucune erreur n'est renvoyée.

POP3_DownLoad

POP3_DownLoad (pop3_ID ; numéroMsg ; enTêteSeul ; nomFichier) -> Résultat

Paramètre	Type		Description
pop3_ID	Entier long	→	Référence d'une connexion POP3
numéroMsg	Entier long	→	Numéro du message
enTêteSeul	Entier	→	0 = Message entier, 1 = En-tête seul
nomFichier	Texte	→	Nom de fichier local
		←	Nom de fichier local utilisé
Résultat	Entier	↩	Code d'erreur

Description

La commande *POP3_DownLoad* permet de télécharger un message d'un serveur POP3 en local sur disque. Tout message POP3 contenant des fichiers joints ou dont la taille est supérieure à 32 Ko devra être téléchargé avec cette commande. Les fichiers joints peuvent être extraits uniquement à partir d'un message préalablement téléchargé.

pop3_ID contient la référence d'une session ouverte avec *POP3_Login*.

Le paramètre *numéroMsg* désigne le numéro du message à récupérer dans la boîte aux lettres. Ce numéro représente la position d'un message dans la liste courante des messages. Attention, le *numéroMsg* d'un message n'est pas une valeur stable, il diffère d'une session à l'autre en fonction des suppressions effectuées.

Le paramètre *enTêteSeul* vous permet de spécifier si vous souhaitez récupérer la totalité du message ou uniquement les informations des en-têtes.

Le paramètre *nomFichier* désigne le nom et/ou l'emplacement du fichier dans lequel vous souhaitez enregistrer le message. Cette valeur peut être spécifiée de trois manières :

""	= Enregistre le fichier dans le dossier défini par *POP3_SetPrefs*, avec le nom "temp1" (si un fichier de ce nom existe déjà, les noms "temp2", "temp3", etc., sont essayés).
"nomFichier"	= Enregistre le fichier dans le dossier défini par *POP3_SetPrefs*, avec le nom *nomFichier.*
"Chemin:nomFichier"	= Enregistre le fichier en utilisant le chemin spécifié par *nomFichier*.

Dans les deux premiers cas, en l'absence de dossier spécifié par *POP3_SetPrefs*, le message est enregistré dans le dossier de la structure de la base de données (avec 4D monoposte) ou dans le dossier de 4D Client (avec 4D Server).

Après l'exécution de la commande, le nom final du fichier est retourné dans le paramètre *nomFichier*. Si vous tentez d'appeler *POP3_Download* avec un *nomFichier* qui existe déjà dans le dossier de téléchargement, ce nom est incrémenté et le nom réellement enregistré sur disque est retourné.

POP3_GetMessage

POP3_GetMessage (pop3_ID ; numéroMsg ; décalage ; longueur ; texteMsg) -> Résultat

Paramètre	Type		Description
pop3_ID	Entier long	➡	Référence d'une connexion POP3
numéroMsg	Entier long	➡	Numéro du message
décalage	Entier long	➡	Caractère à partir duquel commencer la récupération
longueur	Entier long	➡	Nombre de caractères à renvoyer
texteMsg	Texte	⬅	Texte du message
Résultat	Entier	↩	Code d'erreur

Description

La commande *POP3_GetMessage* retourne le contenu du message désigné par *numéroMsg* dans la boîte aux lettres référencée par *pop3_ID*. Sauf spécification contraire de la commande *POP3_SetPrefs*, les caractères Retour à la ligne (Line feed) à l'intérieur du message sont supprimés. La commande *POP3_GetMessage* renvoie l'intégralité du message, y compris les informations des zones d'en-tête.

pop3_ID contient la référence d'une session ouverte avec *POP3_Login*.

Le paramètre *numéroMsg* désigne le message à récupérer dans la boîte aux lettres. Le *numéroMsg* représente la position du message dans la liste courante de messages. Attention, le *numéroMsg* d'un message n'est pas une valeur stable, il diffère d'une session à l'autre en fonction des suppressions effectuées.

Le paramètre *décalage* vous permet d'indiquer la position du caractère (calculée par rapport au début du message) à partir duquel commencer la lecture. Dans la plupart des cas, vous passerez 0 (zéro) dans ce paramètre.

Le paramètre *longueur* indique le nombre de caractères à récupérer au-delà de la position de *décalage*. La longueur maximale de ce paramètre étant limitée à 32 000 caractères pour des raisons historiques, le paramètre *longueur* doit être inférieur à 32 000. Les messages dont la taille est supérieure doivent être récupérés sur disque au moyen de la commande *POP3_Download*.

Le texte récupéré est retourné dans la variable *texteMsg*.

POP3_GetPrefs

POP3_GetPrefs (retoursLigne ; dossierMsg ; dossierDocsJoints) -> Résultat

Paramètre	Type		Description
retoursLigne	Entier	⇐	0 = Ne pas retirer les retours ligne, 1 = Retirer les retours ligne
dossierMsg	Texte	⇐	Chemin d'accès au dossier des messages ("" = aucune modification)
dossierDocsJoints	Texte	⇐	Chemin d'accès du dossier des documents joints ("" = aucune modification)
Résultat	Entier	⇒	Code d'erreur

Description

La commande *POP3_GetPrefs* vous permet de connaître les préférences courantes pour les commandes POP3.

Le paramètre *retoursLigne* retourne le paramétrage courant de l'option de suppression des retours à la ligne.

Le paramètre *dossierMsg* retourne le chemin d'accès local du dossier dans lequel sont enregistrés par défaut les messages récupérés.

Le paramètre *dossierDocsJoints* retourne le chemin d'accès local du dossier dans lequel sont enregistrés par défaut les documents joints extraits des messages.

Note de compatibilité (version 6.8.1) : Le paramètre *dossierDocsJoints* est désormais optionnel. Son usage est déconseillé car il ne sera plus utilisé dans les prochaines versions du plug-in. Notez que ce paramètre n'affecte pas les commandes POP3, il est uniquement utilisé par les commandes MSG.

POP3_Login

POP3_Login (nomServeur ; nomUtilisateur ; motDePasse ; aPOP ; pop3_ID {; paramSession}) -> Résultat

Paramètre	Type		Description
nomServeur	Chaîne	➡	Nom ou adresse IP du serveur de courrier POP3
nomUtilisateur	Chaîne	➡	Nom de l'utilisateur
motDePasse	Chaîne	➡	Mot de passe
aPOP	Entier	➡	0 = Connexion texte clair, 1 = Connexion APOP
pop3_ID	Entier long	⬅	Référence à cette connexion POP3
paramSession	Entier long	➡	1 = Utiliser SSL, 0 ou omis = Ne pas utiliser SSL
Résultat	Entier	↩	Code d'erreur

Description

La commande *POP3_Login* connecte l'utilisateur défini par *nomUtilisateur* et *motDePasse* au serveur de courrier POP3 *nomServeur*. Si *aPOP* vaut 1, le mécanisme APOP (cf. RFC 1321) est utilisé pour la connexion. Si *aPOP* vaut zéro ou est vide, la connexion est effectuée normalement, avec un mot de passe en clair. *POP3_Login* retourne un numéro d'identification pour la connexion (*pop3_ID*) que les commandes ultérieures devront utiliser.

Attention : Les accès aux serveurs POP3 ne sont pas interactifs. Une fois connecté à un serveur, vous devez effectuer les actions nécessaires, puis vous déconnecter dès que possible. Un serveur POP3 déconnecte automatiquement les sessions qui restent inactives pendant un certain laps de temps. Si l'on se réfère à la RFC pour POP3, la période d'inactivité autorisée est d'environ 30 minutes. Cependant, notre expérience montre que la plupart des serveurs déconnectent les clients inactifs après un délai beaucoup plus court.

Chaque commande en interaction avec le serveur POP3 provoque la réinitialisation du compteur d'inactivité. Si le serveur interrompt votre connexion avant que vous ayez appelé *POP3_Logout*, toutes les suppressions que vous avez demandées sont annulées.

Le paramètre *nomServeur* contient le nom ou l'adresse IP du serveur de courrier POP3. Il est généralement conseillé d'utiliser le nom du serveur.

Le paramètre *nomUtilisateur* contient le nom de l'utilisateur du serveur de courrier POP3. Ce nom ne doit pas contenir le nom du domaine. Par exemple, dans le cas de l'adresse "jack@4d.com", le *nomUtilisateur* est "jack".

Le paramètre *motDePasse* est le mot de passe correspondant au *nomUtilisateur* sur le serveur de courrier POP3.

Le paramètre *aPOP* vous permet d'indiquer si le mécanisme APOP doit être utilisé pour la connexion. Passez 1 pour utiliser ce mécanisme APOP, ou 0 (zéro) pour effectuer la connexion par mot de passe en clair. La valeur par défaut est 0.

Le paramètre *pop3_ID* retourne le numéro de référence de la session. Ce numéro sera utilisé par toutes les commandes ultérieures effectuant des actions au sein de cette session.

Le paramètre optionnel *paramSession* vous permet d'activer le protocole SSL pour la connexion :

- si vous passez 1, la connexion au serveur POP3 sera effectuée en SSL (mode synchrone),
- si vous passez 0 ou omettez ce paramètre, la connexion sera effectuée en mode standard non sécurisé.

POP3_Logout

POP3_Logout (pop3_ID) -> Résultat

Paramètre	Type		Description
pop3_ID	Entier long	⇒	Référence d'une connexion POP3
		⇐	0 = Déconnexion réussie
Résultat	Entier	↩	Code d'erreur

Description

La commande *POP3_Logout* referme la session POP3 désignée par *pop3_ID*. Si la déconnexion s'est correctement déroulée, la valeur 0 (zéro) est retournée dans le paramètre *pop3_ID*.

Lorsque vous vous déconnectez d'un serveur POP3, les éventuelles suppressions demandées pendant la session sont effectuées sur le serveur. Si vous voulez annuler ces suppressions, appelez la commande *POP3_Reset* avant *POP3_Logout*.

pop3_ID contient la référence d'une session ouverte avec *POP3_Login*.

POP3_MsgInfo

POP3_MsgInfo (pop3_ID ; numéroMsg ; tailleMsg ; uniqueID) -> Résultat

Paramètre	Type		Description
pop3_ID	Entier long	→	Référence d'une connexion POP3
numéroMsg	Entier long	→	Numéro du message
tailleMsg	Entier long	←	Taille du message
uniqueID	Chaîne	←	ID unique d'un message sur le serveur
Résultat	Entier	↩	Code d'erreur

Description

La commande *POP3_MsgInfo* retourne la taille et l'ID unique du message désigné par *numéroMsg* dans la boîte aux lettres ouverte référencée par *pop3_ID*.

pop3_ID contient la référence d'une session ouverte avec *POP3_Login*.

Le paramètre *numéroMsg* indique le message sur lequel vous souhaitez obtenir des informations. Le *numéroMsg* représente la position du message dans la liste courante des messages. Attention, le *numéroMsg* d'un message n'est pas une valeur stable, il diffère d'une session à l'autre en fonction des suppressions effectuées.

Le paramètre *tailleMsg* retourne la taille du message.

Le paramètre *uniqueID* retourne l'ID unique du message sur le serveur. L'*uniqueID* est une valeur affectée au message par le serveur POP3. Cette valeur ne varie pas d'une session à l'autre, à la différence de *numéroMsg*. La valeur *uniqueID* est une référence utile pour vérifier si la base de données a déjà téléchargé un message depuis le serveur.

POP3_MsgLst

POP3_MsgLst (pop3_ID ; départ ; fin ; tabEnTêtesMsg ; tabNumMsg ; tabIDMsg ; tabValeursMsg) -> Résultat

Paramètre	Type		Description
pop3_ID	Entier long	→	Référence d'une connexion POP3
départ	Entier long	→	Numéro du premier message
fin	Entier long	→	Numéro du dernier message
tabEnTêtesMsg	Tableau chaîne	→	Tableau des en-têtes à récupérer
tabNumMsg	Tableau entier long	←	Tableau des numéros de message
tabIDMsg	Tableau chaîne	←	Tableau alphanumérique des ID uniques
tabValeursMsg	Tableau alpha 2D, Tableau texte 2D	←	Tableau 2D des valeurs des en-têtes
Résultat	Entier	↩	Code d'erreur

Description

La commande *POP3_MsgLst* permet d'obtenir des informations contenues dans les en-têtes d'un ensemble de messages. Vous remplissez le tableau alphanumérique ou texte *tabEnTêtesMsg* avec les noms des en-têtes que vous souhaitez récupérer. Le tableau 2D *tabValeursMsg* reçoit le contenu des en-têtes. A chaque en-tête demandé correspond une "ligne" du tableau *tabValeursMsg*.

La commande *POP3_MsgLst* ne peut renvoyer que le contenu des en-têtes, elle ne peut pas servir à récupérer le corps d'un message.

Note : Les champs d'en-tête étant susceptibles de contenir des caractères étendus, vous pouvez automatiser la gestion de ceux-ci à l'aide de la commande *POP3_Charset*.

Exemple

```
aEnTêtes{1}:="Date:"
aEnTêtes{2}:="From:"
aEnTêtes{3}:="Subject:"
POP3_MsgLst(◇POP3_ID;vPremier;vDernier;aEnTêtes;aNumMsg;aUID;aValeurs)
aValeurs{1}{1}  ` par exemple "Jeudi 19 novembre 1998, 00:24:02 -0800"
aValeurs{2}{1}  ` par exemple "Jack@4d.com"
aValeurs{3}{1}  ` par exemple "Appelez votre femme"
```

Les erreurs sont gérées de la façon suivante :

1) Seules les erreurs relatives à la communication sont renvoyées. Si la commande ne peut pas achever sa tâche en raison d'une erreur (réseau, syntaxe, serveur, etc.), le code d'erreur approprié est renvoyé.

2) Si un message appartenant à l'intervalle spécifié n'existe pas ou comporte une erreur :
- Aucun élément de tableau n'est créé pour ce message.
- Aucune erreur n'est renvoyée.

3) L'incapacité à localiser un ou plusieurs en-têtes dans un message ne constitue pas une erreur :
- Un élément de tableau est créé pour le message.
- Les éléments de tableau "numéro" et "ID" contiennent les valeurs appropriées.
- Pour chaque en-tête introuvable dans le message, une chaîne vide ("") est renvoyée à l'élément de tableau.
- Aucun code d'erreur n'est renvoyé.

Note : Si le paramètre *premierMsg* est supérieur au paramètre *dernierMsg*, cette commande ne fait rien. Aucune erreur n'est renvoyée.

POP3_MsgLstInfo

POP3_MsgLstInfo (pop3_ID ; premierMsg ; dernierMsg ; tabTaillesMsg ; tabNumMsg ; tabIDMsg) -> Résultat

Paramètre	Type		Description
pop3_ID	Entier long	→	Référence d'une connexion POP3
premierMsg	Entier long	→	Numéro du premier message
dernierMsg	Entier long	→	Numéro du dernier message
tabTaillesMsg	Tableau entier long	←	Tableau des tailles
tabNumMsg	Tableau entier long	←	Tableau des numéros de message
tabIDMsg	Tableau chaîne	←	Tableau des ID uniques
Résultat	Entier	↩	Code d'erreur

Description

La commande *POP3_MsgLstInfo* retourne la taille, le numéro et l'ID unique d'un ensemble de messages contenus dans la boîte aux lettres. Ces informations sont renvoyées dans trois tableaux, chaque élément de tableau correspondant à un message.

La commande *POP3_MsgLstInfo* ajuste la taille de chaque tableau au nombre de messages récupérés. Vous devez toutefois déclarer les tableaux avant d'appeler la commande.

La commande *POP3_MsgLstInfo* ne renvoie pas d'erreur si elle ne parvient pas à récupérer des informations sur un message particulier. Dans ce cas, la commande ignore le message et ne crée pas d'élément correspondant dans les tableaux. Par conséquent, si la commande réussit à lire tous les messages, le tableau *tabNumMsg* contiendra une suite continue de valeurs numériques séquentielles. En cas de problème, des numéros seront manquants dans la suite numérique de *tabNumMsg*.

pop3_ID contient la référence d'une session ouverte avec *POP3_Login*.

Le paramètre *premierMsg* indique le numéro du premier message à examiner.
Le paramètre *dernierMsg* indique le numéro du dernier message à examiner.
Ce numéro représente la position d'un message dans la liste des messages de la boîte aux lettres identifiée par *pop3_ID*.

Le tableau *tabTaillesMsg* reçoit la taille de chaque message dont le numéro est compris entre *premierMsg* et *dernierMsg*.

Le tableau *tabNumMsg* reçoit les numéros de chaque message.

Le tableau *tabIDMsg* reçoit les ID uniques de chaque message.

Note : Si le paramètre *premierMsg* est supérieur au paramètre *dernierMsg*, cette commande ne fait rien. Aucune erreur n'est renvoyée.

POP3_Reset

POP3_Reset (pop3_ID) -> Résultat

Paramètre	Type		Description
pop3_ID	Entier long	⇒	Référence d'une connexion POP3
Résultat	Entier	⇐	Code d'erreur

Description

La commande *POP3_Reset* permet d'annuler l'option "à supprimer" appliquée aux messages durant la session courante.

Note : La commande *POP3_Delete* ne supprime pas immédiatement les messages, son action consiste uniquement à marquer les messages comme étant "à supprimer". Les messages ne sont réellement supprimés sur le serveur POP3 que si la déconnexion (*POP3_Logout*) a été correctement effectuée.

pop3_ID contient la référence d'une session ouverte avec *POP3_Login*.

POP3_SetPrefs

POP3_SetPrefs (retoursLigne ; dossierMsg ; dossierDocsJoints) -> Résultat

Paramètre	Type		Description
retoursLigne	Entier	➡	0 = Ne pas retirer les retours à la ligne, 1 = Retirer les retours à la ligne, -1 = Aucune modification
dossierMsg	Texte	➡	Chemin d'accès au dossier des messages ("" = aucune modification)
dossierDocsJoints	Texte	➡	Chemin d'accès du dossier des documents joints ("" = aucune modification)
Résultat	Entier	↩	Code d'erreur

Description

La commande *POP3_SetPrefs* définit des préférences générales pour toutes les commandes POP3 ultérieures.

Le paramètre *retoursLigne* vous permet de préciser comment traiter les caractères de retour à la ligne dans les messages enregistrés. La plupart des serveurs POP3 associent un caractère Retour chariot (carriage return) et un caractère Retour à la ligne (line feed) pour indiquer la fin d'une ligne, à la différence des applications Macintosh qui requièrent un simple Retour chariot. Dans ce cas, cette option vous permet de supprimer les caractères Retour à la ligne superflus du texte des messages.

- Si vous passez 0 (zéro) les messages récupérés seront conservés dans le format du serveur POP3.
- Si vous passez 1, les caractères Retour à la ligne seront supprimés des messages récupérés.
- Si vous passez -1, l'option reste telle qu'elle était précédemment définie.

Par défaut, cette option a pour valeur 1, les retours à la ligne rencontrés dans les messages sont automatiquement supprimés.

Le paramètre *dossierMsg* indique le chemin d'accès local du dossier dans lequel les messages récupérés à l'aide de la commande *POP3_Download* doivent être enregistrés par défaut.

Note de compatibilité (version 6.8.1) : Les paramètres *retoursLigne* et *dossierMsg* étaient précédemment utilisés par les commandes MSG (thème **IC Downloaded Mail**). Ce n'est désormais plus le cas lorsque la commande *MSG_SetPrefs* est utilisée.

Le paramètre *dossierDocsJoints* indique le chemin d'accès local du dossier dans lequel les fichiers joints doivent être enregistrés lorsque la commande *MSG_Extract* extrait les documents joints du corps du message.

Note de compatibilité (version 6.8.1) : Le paramètre *dossierDocsJoints* est également présent dans la commande *MSG_SetPrefs*, par conséquent vous pouvez le fixer à l'aide de l'une de ces deux commandes. L'usage de la commande *MSG_SetPrefs* est fortement recommandé. Le paramètre de la commande *POP3_SetPrefs*, conservé pour des raisons de compatibilité, ne sera plus utilisé dans les prochaines versions du plug-in. Ce paramètre est désormais optionnel et son usage est déconseillé. Cette recommandation s'applique également à la commande *POP3_GetPrefs*.

POP3_UIDToNum

POP3_UIDToNum (pop3_ID ; uniqueID ; numéroMsg) -> Résultat

Paramètre	Type		Description
pop3_ID	Entier long	➡	Référence à une connexion POP3
uniqueID	Chaîne	➡	ID unique d'un message sur le serveur
numéroMsg	Entier long	⬅	Numéro du message
Résultat	Entier	↩	Code d'erreur

Description

La commande *POP3_UIDToNum* retourne le numéro de message *numéroMsg* correspondant au message désigné par *uniqueID* dans la boîte aux lettres référencée par *pop3_ID*.
Le *numéroMsg* d'un message étant une valeur fluctuante, relative aux autres messages de la liste, cette commande renvoie la position courante d'un message dont les informations peuvent avoir été récupérées à l'occasion d'une précédente session POP3.

pop3_ID contient la référence d'une session ouverte avec *POP3_Login*.

Le paramètre *uniqueID* contient l'ID unique du message à localiser sur le serveur POP3. La commande recherche cette valeur dans les en-têtes des messages référencés dans la session identifiée par *pop3_ID*. Une fois trouvé, le numéro du message est retourné dans *numéroMsg*.

Le paramètre *numéroMsg* retourne le numéro courant du message désigné par *uniqueID*. Si l'*uniqueID* ne peut pas être trouvé sur le serveur, *numéroMsg* retourne 0 (zéro), aucune erreur n'est renvoyée.

POP3_VerifyID

POP3_VerifyID (pop3_ID) -> Résultat

Paramètre	Type		Description
pop3_ID	Entier long	⇒	Référence à une connexion POP3
		⇐	0 = La connexion est déjà close
Résultat	Entier	↩	Code d'erreur

Description

Un serveur POP3 déconnecte automatiquement les comptes demeurés inactifs pendant une certaine période, déterminée par son administrateur (ce mécanisme est appelé timeout).
Chaque commande en interaction avec le serveur POP3 provoque la remise à zéro du compteur d'inactivité.
La commande *POP3_VerifyID* provoque également la remise à zéro du compteur d'inactivité de la session POP3 spécifiée. Elle permet de conserver une session active en cas de risque de dépassement du délai.

Lors de son exécution, la commande *POP3_VerifyID* vérifie que la connexion n'a pas déjà été close.
Si la session est toujours ouverte, la commande indique au serveur POP3 de remettre à zéro le compteur d'inactivité pour la session.
Si la connexion a déjà été fermée, *POP3_VerifyID* retourne l'erreur appropriée, libère la mémoire utilisée par la session POP3 et renvoie 0 dans le paramètre *pop3_ID*.

Le paramètre *pop3_ID* contient une référence de session ouverte créée avec *POP3_Login*.

IC Send Mail

- Envoi de courrier, Présentation
- SMTP_AddHeader
- SMTP_Attachment
- SMTP_Auth
- SMTP_Bcc
- SMTP_Body
- SMTP_Cc
- SMTP_Charset
- SMTP_Clear
- SMTP_Comments
- SMTP_Date
- SMTP_Encrypted
- SMTP_From
- SMTP_GetPrefs
- SMTP_Host
- SMTP_InReplyTo
- SMTP_Keywords
- SMTP_MessageID
- SMTP_New
- SMTP_QuickSend
- SMTP_References
- SMTP_ReplyTo
- SMTP_Send
- SMTP_Sender
- SMTP_SetPrefs
- SMTP_Subject
- SMTP_To

Envoi de courrier, Présentation

SMTP (Simple Mail Transport Protocol) est le protocole standard du courrier électronique sur Internet. Avec les commandes Internet de 4D, vous pouvez créer des messages électroniques simples au moyen d'une seule commande, ou des messages complexes grâce à une série de commandes. Les commandes SMTP vous permettent de contrôler toutes les parties d'un message électronique, dont les en-têtes de Réponse à (Reply-To) et d'Expéditeur (Sender), les pièces jointes, les commentaires et les références.

L'intégration des commandes Internet dans 4D vous permet de développer des bases de données très puissantes pouvant envoyer des messages et des documents sur Internet. A l'aide des commandes SMTP, vous pourrez notamment :

- automatiser l'envoi d'états ou de documents créés dans 4D,
- construire des applications 4D informant automatiquement les développeurs lors d'événements spécifiques, par exemple en cas d'incident (en utilisant **APPELER SUR ERREUR**),
- envoyer des mailings automatiquement à des personnes dans le monde entier.

Ces commandes, associées aux commandes POP3 (récupération de fichiers et de documents joints), FTP et TCP, fournissent au développeur les outils lui permettant d'augmenter de façon considérable les capacités de communications de ses bases de données 4D.

Deux méthodes de création d'un message électronique

Les commandes SMTP permettent d'envoyer du courrier électronique de deux manières distinctes, appelées précédemment méthode "simple" et méthode "complexe". La méthode "simple" implique une seule commande, *SMTP_QuickSend*, qui accepte tous les paramètres nécessaires à l'adressage et à l'envoi d'un message.

La majorité des messages envoyés dans le monde sont plutôt simples dans leur construction : quelqu'un "ici" souhaite envoyer un "message" quelconque à quelqu'un se trouvant "là-bas" à propos d'un "objet" spécifique. S'il s'agissait d'un courrier classique, vous devriez écrire le texte, fermer l'enveloppe, rédiger l'adresse puis porter la lettre au bureau de poste pour qu'elle soit expédiée. Avec *SMTP_QuickSend*, vous pouvez préciser l'Emetteur (From), le Destinataire (To), l'Objet (Subject) et le Corps du message (Message Body) au moyen d'une seule commande.

Cependant, certains messages peuvent nécessiter des paramétrages plus complexes. Par exemple, supposons que la lettre mentionnée plus haut doive être envoyée en plusieurs exemplaires à d'autres destinataires ou qu'un document comme votre rapport annuel doive lui être joint. Dans ce cas, vous devriez photocopier la lettre, imprimer des rapports et préparer une enveloppe pour chaque destinataire. Les commandes SMTP de 4D simplifient la distribution en vous permettant de contrôler tous les aspects de la transmission de courrier électronique. Les documents joints multiples, l'envoi de copie conforme (carbon copy) et de copie discrète (blind carbon) ainsi que toute spécification d'en-tête de courrier peuvent être gérés.

Comprendre la distribution du courrier

L'un des concepts clés pour la compréhension du fonctionnement des commandes SMTP se rapporte au mode de distribution du courrier à ses destinataires. Les commandes SMTP ne transmettent pas directement le courrier à chaque destinataire. Elles effectuent la composition et le formatage appropriés au courrier et transmettent le résultats au serveur SMTP spécifié par la commande *SMTP_Host*. Le serveur SMTP est généralement une machine se trouvant dans votre entreprise ou chez votre fournisseur d'accès à Internet. Le serveur SMTP détermine alors le chemin de transmission optimal pour votre courrier et programme sa distribution en fonction des paramètres configurés par l'administrateur du courrier.

Conditions minimales requises pour envoyer un message SMTP complexe

Pour une bonne transmission des messages électroniques via SMTP, les commandes doivent avoir été correctement paramétrées. La séquence de commandes suivante représente le minimum requis afin que la transmission de courrier puisse aboutir :

- *SMTP_New*
  Réserve un espace mémoire pour le nouveau message et fournit une référence devant être utilisée par

les commandes ultérieures.

- *SMTP_Host*
  Spécifie le serveur SMTP auquel le message doit être transmis.
- *SMTP_From*
  Une adresse au moins doit être spécifiée dans l'en-tête "From".
- *SMTP_To*
  Une adresse au moins doit être spécifiée dans l'en-tête "To".
- *SMTP_Send*
  Envoie le message.
- *SMTP_Clear*
  Réinitialise (libère) l'espace mémoire réservé pour le message.

Si seules ces commandes sont exécutées, un message ne contenant pas d'objet ni de corps est envoyé. Il est donc nécessaire de définir des informations complémentaires pour envoyer un message.

Unicode par défaut (4D v14)

A compter de 4D v14, les champs "objet" (*subject*) et "corps" (*body*) des commandes SMTP utilisent par défaut le jeu de caractères UTF-8. Ce jeu de caractères sera correctement interprété par la quasi totalité des clients de messagerie. Ce fonctionnement simplifie largement l'utilisation des commandes SMTP et limite désormais l'utilité des commandes **SMTP_Charset** et **SMTP_SetPrefs**.

SMTP_AddHeader

SMTP_AddHeader (smtp_ID ; nomEnTête ; texteEnTête {; supprimerOption}) -> Résultat

Paramètre	Type		Description
smtp_ID	Entier long	➡	Référence de message
nomEnTête	Chaîne	➡	Nom de l'en-tête
texteEnTête	Texte	➡	Texte de l'en-tête
supprimerOption	Entier	➡	0 = Ajouter 1 = Remplacer tous les en-têtes par 'nomEnTête', 2 = Supprimer tous les en-têtes nommés 'nomEnTête'
Résultat	Entier	↩	Code d'erreur

Description

La commande *SMTP_AddHeader* vous permet d'ajouter votre propre en-tête au message référencé par *smtp_ID*. Outre les en-têtes standard gérés par défaut par les commandes Internet de 4D, il existe deux catégories d'en-têtes supplémentaires : les en-têtes "utilisateur" (User-defined) et les en-têtes "étendus" (Extended). La commande *SMTP_AddHeader* permet d'ajouter à la fois un nouvel en-tête et les données qu'il contient.

En-têtes "étendus" : Ces en-têtes ont été officiellement reconnus par le NIC et ont été définis après les spécifications SMTP initiales. Ces en-têtes ont généralement un rôle spécifique affectant le comportement de différents logiciels. Les en-têtes "étendus" ne commencent jamais par la lettre "X".

En-têtes "utilisateur" : Le protocole SMTP permet à quiconque de créer ses propres définitions d'en-tête. Tous les en-têtes définis par l'utilisateur doivent commencer par les caractères "X-" pour éviter tout conflit avec un futur en-tête "étendu". Les en-têtes "utilisateur" sont particulièrement utiles lorsque vous contrôlez les deux extrémités des communications.

Les en-têtes "utilisateur" permettent au développeur de stocker des données qui peuvent être facilement extraites au moyen de la commande *MSG_FindHeader*. Par exemple, vous pouvez créer un en-tête nommé "X-001001", qui contient la valeur du champ 01 de la table 01. Il est possible d'ajouter un nombre illimité d'en-têtes à un message. Les en-têtes "utilisateur" permettent d'insérer des informations plus faciles à extraire que celles nécessitant l'analyse du corps du message.

smtp_ID contient l'identifiant d'un message électronique créé avec la commande *SMTP_New*.

Le paramètre *nomEnTête* contient le nom de l'en-tête à ajouter.

Le paramètre *texteEnTête* contient les informations à affecter à la zone d'en-tête identifiée par *nomEnTête*.

Attention : Le texte ne doit pas contenir de retours à la ligne (code ASCII=10). Un retour à la ligne désigne la fin de la section d'en-tête et le début du corps du texte. Les en-têtes suivants risquent alors d'être considérés comme le corps du texte et de ne pas être correctement reconnus par le logiciel du serveur ou du client. Pour plus d'informations sur les zones d'en-tête, veuillez consulter la RFC#822.

Note : La commande ne fait rien si *nomEnTête* ou *texteEnTête* est une chaîne vide (aucun en-tête n'est ajouté).

Le paramètre *supprimerOption* vous permet de préciser s'il faut ou non supprimer l'en-tête courant.

- Si vous passez 0 (zéro), l'en-tête *nomEnTête* est ajouté au message.
- Si vous passez 1, tous les en-têtes du message sont remplacés par l'en-tête *nomEnTête*. Dans ce cas, si *nomEnTête* contient une chaîne vide, tous les en-têtes du message sont supprimés.
- Si vous passez 2, tous les en-têtes *nomEnTête* sont supprimés du message.

Note : A compter de la version 14 de 4D Internet Commands, si vous souhaitez envoyer un message au format HTML, il n'est plus nécessaire de modifier l'en-tête "Content-Type" à l'aide de **SMTP_AddHeader**. Vous pouvez déclarer l'utilisation du format HTML directement à l'aide de la commande **SMTP_Body**, auquel cas le "Content-Type" sera automatiquement défini comme "text/html;charset=utf-8" (sinon, le "Content-Type" est défini par défaut comme "text/plain;charset=utf-8"). Toutefois, pour des besoins spécifiques, vous pouvez toujours "forcer" le champ "Content-Type" avec **SMTP_AddHeader**. Dans ce cas, veillez à bien spécifier le charset (en principe "charset=utf-8" car, par défaut 4D IC envoie toujours le body en UTF-8).

SMTP_Attachment

SMTP_Attachment (smtp_ID ; nomFichier ; typeEncodage ; supprimerOption {; idAttachment {; contentType}}) -> Résultat

Paramètre	Type		Description
smtp_ID	Entier long	⇒	Référence de message
nomFichier	Texte	⇒	Nom du fichier à joindre
typeEncodage	Entier	⇒	Mac & Win: 0 = Pas d'encodage (n'envoie que la DataFork) ±1 = BinHex ±2 = Base64 (n'envoie que la DataFork) ±7 = UUEncode; Mac seulement: ±3 = AppleSingle ±4 = AppleDouble ±5 = AppleSingle ET Base64 ±6 = AppleDouble ET Base64
supprimerOption	Entier	⇒	0 = Ajouter à la liste existante, 1 = Remplacer toutes les pièces jointes par nomFichier, 2 = Ne supprimer que cette pièce jointe
idAttachment	Texte	⇒	Identifiant de fichier joint (messages en HTML uniquement)
contentType	Texte	⇒	Valeur de type de contenu à définir
Résultat	Entier	⇐	Code d'erreur

Description

La commande **SMTP_Attachment** vous permet de joindre des fichiers de type binaire ou texte à votre message dans le format MIME. Cette commande peut être appelée plusieurs fois pour joindre plusieurs documents à un message. Si vous passez une valeur positive dans le paramètre *typeEncodage*, l'encodage sera effectué au moment de l'envoi du message.

smtp_ID contient l'identifiant d'un message créé avec la commande *SMTP_New*.

Le paramètre *nomFichier* désigne le fichier que vous voulez joindre au message. Cette valeur peut être spécifiée de deux manières :

"nomFichier"	= Recherche le nom du fichier dans le répertoire de la structure de la base de données.
"Chemin:nomFichier"	= Chemin d'accès complet (nom du fichier compris).

Note : Dans 4D Internet Commands v16 R2 et suivantes, la taille des noms de fichiers manipulés par les routines est limitée à 1024 caractères, quel que soit l'OS. Cependant, dans les versions macOS 32 bits, 4D Internet Commands doit utiliser des APIs de gestion de fichiers déclarées obsolètes et non maintenues par Apple, Inc. Dans cet environnement, suivant la version de macOS, les limitations relatives à la longueur des noms de fichiers (ainsi qu'aux caractères qu'ils acceptent) peuvent être plus restrictives. Nous recommandons fortement de mettre à niveau 4D Internet Commands en version 64 bits dès que possible.

Le paramètre *typeEncodage* indique le type d'encodage à appliquer au fichier avant de l'intégrer au message. S'il s'agit d'un fichier binaire, vous devez utiliser un type d'encodage approprié (BinHex, AppleSingle). L'encodage le plus courant est BinHex.

Si la valeur de *typeEncodage* est positive, la commande encode automatiquement le fichier au moment de l'envoi du message à l'aide de la commande *SMTP_Send*.
Si le fichier est volumineux, l'exécution de la commande *SMTP_Send* peut durer quelques instants. Il est possible de gagner du temps lorsqu'un même fichier est envoyé plusieurs fois : l'astuce consiste à encoder le fichier une seule fois avec la commande *IT_Encode*, puis de le joindre au message en passant une valeur négative dans *typeEncodage*. Lorsque le paramètre *typeEncodage* reçoit une valeur négative, aucun encodage supplémentaire n'est effectué, mais le type d'encodage est décrit dans l'en-tête du message du fichier joint, de manière à ce que le logiciel de messagerie du destinataire puisse l'interpréter correctement.

Note : Vous ne devez pas passer d'élément de tableau dans le paramètre *typeEncodage*.

Le paramètre *supprimerOption* indique comment traiter le fichier joint.

- Si vous passez 0 (zéro), le fichier joint est ajouté à la liste courante des fichiers joints.
- Si vous passez 1, *nomFichier* remplace tous les fichiers éventuellement déjà joints au message. Dans ce cas, si *nomFichier* est une chaîne vide, tous les fichiers joints sont supprimés.
- Si vous passez 2, le fichier désigné par *nomFichier* est supprimé de la liste des fichiers joints.

Le paramètre *idAttachement* vous permet d'associer le fichier joint à une référence définie dans le corps du message via la balise HTML **<img src=\"cid:ID\">**. Ce principe permet d'afficher le contenu du fichier, par exemple une image, à l'intérieur du message sur le client de messagerie.
Ce fonctionnement est pris en charge uniquement avec les messages formatés en HTML. A noter également que le rendu final pourra varier en fonction du client de messagerie.

Le paramètre optionnel *contentType* vous permet de définir explicitement le type de contenu du fichier joint. Par défaut, si ce paramètre est omis ou contient une chaîne vide, 4DIC définit automatiquement le type de contenu du fichier joint sur la base de son extension. Les règles suivantes sont appliquées :

Extension	Type de contenu
jpg, jpeg	image/jpeg
png	image/png
gif	image/gif
pdf	application/pdf
doc	application/msword
xls	application/vnd.ms-excel
ppt	application/vnd.ms-powerpoint
zip	application/zip
gz	application/gzip
json	application/json
js	application/javascript
ps	application/postscript
xml	application/xml
htm, html	text/html
mp3	audio/mpeg
other	application/octet-stream

Dans *contentType*, vous pouvez passer une chaîne spécifiant le type de contenu pour le fichier (type MIME), par exemple "video/mpeg". Cette valeur de content-type sera alors utilisée pour le fichier joint, quelle que soit son extension.

Note : Passez une chaîne vide ("") dans le paramètre *idAttachment* si vous ne souhaitez pas l'utiliser.

Exemple 1

Envoi d'un message en HTML avec une image incluse :

```
$erreur:=SMTP_New($smtp_id)
$erreur:=SMTP_Host($smtp_id;"smtp.gmail.com")
$erreur:=SMTP_From($smtp_id;"henry@gmail.com")
$erreur:=SMTP_ReplyTo($smtp_id;"replies@gmail.com")
$erreur:=SMTP_Subject($smtp_id;"Test HTML  &  image incluse")
$erreur:=SMTP_To($smtp_id;"jean@4d.com";1)
$erreur:=SMTP_Body($smtp_id;"<html><B><I>Hello world in bold!</I></B> <img src=\"cid:myID123\">(normal
text)</html>";4)
$erreur:=SMTP_Attachment($smtp_id;"c:\\temp\\tulips.jpg";2;0;"myID123")
$erreur:=SMTP_Auth($smtp_id;"henry@gmail.com";"*******")
$erreur:=SMTP_Send($smtp_id;1)
$erreur:=SMTP_Clear($smtp_id)
```

Exemple 2

Vous souhaitez déclarer vos fichiers de configuration (settings) en tant que fichiers XML :

```
$path:=Get 4D folder(Database folder)+"Settings.mySettings"
$err:=SMTP_Attachment($smtp_id;$path;2;0;"myID123";"application/xml")
```

SMTP_Auth

SMTP_Auth (smtp_ID ; nomUtilisateur ; motDePasse {; authMode}) -> Résultat

Paramètre	Type		Description
smtp_ID	Entier long	➡	Référence de message
nomUtilisateur	Chaîne	➡	Nom d'utilisateur pour l'authentification SMTP
motDePasse	Chaîne	➡	Mot de passe pour l'authentification SMTP
authMode	Entier	➡	Mode d'authentification à utiliser : 0 ou omis=Mode défini par le serveur, 1=PLAIN, 2=LOGIN, 3=CRAM-MD5
Résultat	Entier	↩	Code d'erreur

Description

La commande *SMTP_Auth* permet l'envoi du message référencé par *smtp_ID* lorsqu'un mécanisme d'authentification est requis par le serveur SMTP. L'authentification peut être requise par certains serveurs SMTP afin de réduire le risque que des messages soient falsifiés ou que l'identité des émetteurs soit usurpée, notamment à des fins de spamming.

Cette commande peut être utilisée dans votre code même si l'authentification n'est pas nécessaire car elle n'est exécutée que si les paramètres *nomUtilisateur* et *motDePasse* ne sont pas des chaînes vides.

smtp_ID contient l'identifiant d'un message électronique créé avec la commande *SMTP_New*.

nomUtilisateur contient le nom du compte utilisateur défini sur le serveur SMTP. Ce paramètre ne doit pas inclure le nom de domaine. Par exemple, pour l'adresse "jack@4d.com", le paramètre *nomUtilisateur* doit uniquement contenir "jack".

motDePasse contient le mot de passe associé au compte *nomUtilisateur* sur le serveur SMTP.

Note : Si *nomUtilisateur* et/ou *motDePasse* contiennent des chaînes vides, la commande *SMTP_Auth* n'est pas exécutée.

Le paramètre facultatif *authMode* permet de "forcer" le mode d'authentification utilisé. Vous pouvez passez 0, 1, 2 ou 3 dans ce paramètre :

- si vous passez 0 (zéro) le mode d'authentification utilisé par la commande *SMTP_Auth* sera le mode le plus sécurisé pris en charge par le serveur (CRAM-MD5, LOGIN puis PLAIN),
- si vous passez 1, la méthode d'authentification utilisée sera PLAIN,
- si vous passez 2, la méthode d'authentification utilisée sera LOGIN,
- si vous passez 3, la méthode d'authentification utilisée sera CRAM-MD5.
  Si *authMode* est omis, par défaut la valeur 0 est utilisée. Si la méthode d'authentification demandée par ce paramètre n'est pas gérée par le serveur SMTP, une erreur est retournée.

Exemple

Cet exemple permet d'envoyer un message avec ou sans authentification, en fonction du contenu de champs spécifiques stockés dans la base 4D :

```
C_LONGINT($vErreur)
C_LONGINT($vSmtp_id)
C_TEXT($vAuthUtilisateur;$vAuthMotPasse)

$vErreur:=SMTP_New($vSmtp_id)
$vErreur:=SMTP_Host($vSmtp_id;"wkrp.com")
$vErreur:=SMTP_From($vSmtp_id;"herb_tarlick@wkrp.com")
$vErreur:=SMTP_Subject($vSmtp_id;"Es-tu disponible ?")
$vErreur:=SMTP_To($vSmtp_id;"Dupont@wkrp.com")
$vErreur:=SMTP_Body($vSmtp_id;"Peut-on se voir ?")

` Les champs sont saisis si le serveur utilise un mécanisme
` d'authentification. Else, des chaînes vides sont retournées
$vAuthUtilisateur:=[Compte]AuthUtil
$vAuthMotPasse:=[Compte]AuthPass
```

```
$vErreur:=SMTP_Auth($vSmtp_id;$vAuthUtilisateur;$vAuthMotPasse)
$vErreur:=SMTP_Send($vSmtp_id)
$vErreur:=SMTP_Clear($vSmtp_id)
```

SMTP_Bcc

SMTP_Bcc (smtp_ID ; copieDiscrète {; supprimerOption}) -> Résultat

Paramètre	Type		Description
smtp_ID	Entier long	➡	Référence de message
copieDiscrète	Texte	➡	Liste d'adresses
supprimerOption	Entier	➡	0 = Ajouter, 1 = Remplacer, 2 = Supprimer
Résultat	Entier	↩	Code d'erreur

Description

La commande *SMTP_Bcc* ajoute des destinataires en "copie discrète" (Blind carbon copy) au message spécifié par *smtp_ID*. Pour envoyer un message, le champ "Bcc" n'est pas obligatoire.

La seule façon de préserver la confidentialité de la liste d'adresses lors de l'envoi de courrier à un ensemble de personnes est d'insérer ces adresses dans la zone d'en-tête "Bcc". Les adresses listées dans cet en-tête ne sont pas envoyées dans le corps ni dans l'en-tête du message. Ces adresses ne peuvent être lues par aucun destinataire du message.

Un destinataire "Bcc" peut visualiser tous les destinataires "A" et "Cc", mais pas les autres destinataires "Bcc". Généralement, lors d'envoi de courrier en masse, tous les destinataires doivent être placés dans l'en-tête "Bcc". Cela permet d'éviter que les messages reçus soient encombrés par une grande liste d'adresses, et de ne pas communiquer les adresses des destinataires.

Une autre raison de l'utilisation de "Bcc" est que la plupart des messageries électroniques disposent d'une fonction "Répondre à tous". Celle-ci transfère toutes les adresses présentes dans les champs "A" et "Cc" du message reçu dans le champ "A" du message à renvoyer. Placer les adresses dans l'en-tête "Bcc" permet d'éviter qu'un ou plusieurs destinataires répondent à tous les destinataires initiaux.

smtp_ID contient l'identifiant d'un message électronique créé avec la commande *SMTP_New*.

Le paramètre *copieDiscrète* contient une ou plusieurs adresses.

Le paramètre *supprimerOption* vous permet de préciser s'il faut conserver ou supprimer la zone d'en-tête "Cc" éventuellement existante :

- Si vous passez 0 (zéro), le contenu du paramètre passé est ajouté au contenu de l'en-tête existant.
- Si vous passez 1, le contenu du paramètre passé remplace le contenu de l'en-tête existant. Dans ce cas, si vous avez passé une chaîne vide dans *copieDiscrète*, l'en-tête "Bcc" est supprimé.
- Si vous passez 2, l'en-tête "Bcc" est supprimé du message.
- Si *supprimerOption* est omis, par défaut la valeur 0 est utilisée.

Exemple

Dans cet exemple, pour chaque enregistrement de la table [Personnes], une adresse est ajoutée à la liste de copies discrètes :

```
$erreur:=SMTP_From($smtp_id;"sales@massmarket.com")
$erreur:=SMTP_Subject($smtp_id;"Ventes incroyables ! Seulement cette semaine !")
$erreur:=SMTP_Body($smtp_id;$CorpsGénérique)
For($i;1;Records in selection([Personnes]))
   $erreur:=SMTP_Bcc($smtp_id;[Personnes]Email;0)  `Ajoute cette adresse e-mail à la liste BCC
   NEXT RECORD([Personnes])
End for
$erreur:=SMTP_Send($smtp_id)  `Envoie le message à tout le monde
$erreur:=SMTP_Clear($smtp_id)
```

SMTP_Body

SMTP_Body (smtp_ID ; msgCorps {; option}) -> Résultat

Paramètre	Type		Description
smtp_ID	Entier long	➡	Référence de message
msgCorps	Texte	➡	Corps du message (UTF-8 par défaut)
option	Entier	➡	0 = Remplacer sauf si msgCorps vide, 1 = Remplacer, 2 = Ajouter, 4 = Format HTML (texte brut par défaut)
Résultat	Entier	↩	Code d'erreur

Description

La commande **SMTP_Body** insère le texte de *msgCorps* dans le corps principal du message identifié par *smtp_ID*. Le *msgCorps* est le principal bloc de texte.

smtp_ID contient l'identifiant d'un message électronique créé avec la commande **SMTP_New**.

Le paramètre *msgCorps* contient le corps du message. Par défaut, le corps du message est encodé en UTF-8, ce qui garantit que les caractères envoyés seront correctement interprétés par la quasi totalité des clients de messagerie. Si vous souhaitez utiliser un jeu de caractères spécifique, reportez-vous aux commandes **SMTP_SetPrefs** et **SMTP_Charset**.

Le paramètre *option* vous permet de gérer la concaténation de plusieurs corps et de modifier le format du message (texte ou HTML) :

- Si vous passez 0 (zéro), le contenu du paramètre *msgCorps* remplace le corps de message éventuellement présent, sauf si vous passez une chaîne vide dans *msgCorps*, auquel cas le corps de message existant est conservé.
- Si vous passez 1, le contenu du paramètre *msgCorps* remplace le corps de message éventuellement présent. Dans ce cas, si vous passez une chaîne vide dans *msgCorps*, le corps du message est supprimé du message.
- Si vous passez 2, le contenu de *msgCorps* est ajouté au corps du message, à la suite de tout texte déjà défini (concaténation).
- Si vous passez 4, vous indiquez que le contenu de *msgCorps* est du HTML (par défaut, *msgCorps* est envoyé en texte brut).
  Si *option* est omis, par défaut la valeur 0 est utilisée.

Pour combiner l'envoi du message en HTML et une option de remplacement, il suffit d'additionner les valeurs. Par exemple, passez 1+4 dans *option* afin de remplacer le corps et de l'envoyer en HTML.

Exemple

Voici un exemple SMTP complet :

```
C_LONGINT($SMTP_ID)
C_BOOLEAN($EnvoyeOK;$OK)
$EnvoyeOK:=False  `Indicateur précisant s'il est applicable à toutes les commandes
Case of
   :(Not(VérifErreur("SMTP_New";SMTP_New($SMTP_ID))))
   :(Not(VérifErreur("SMTP_Host";SMTP_Host($SMTP_ID;◇pref_Server))))
   :(Not(VérifErreur("SMTP_From";SMTP_From($SMTP_ID;vDe))))
   :(Not(VérifErreur("SMTP_To";SMTP_To($SMTP_ID;vA))))
   :(Not(VérifErreur("SMTP_Cc";SMTP_Cc($SMTP_ID;vCC))))
   :(Not(VérifErreur("SMTP_Bcc";SMTP_Bcc($SMTP_ID;vBcc))))
   :(Not(VérifErreur("SMTP_Subject";SMTP_Subject($SMTP_ID;vSujet))))
   :(Not(VérifErreur("SMTP_Comments";SMTP_Comments($SMTP_ID;"Envoyé via 4D"))))
   :(Not(VérifErreur("SMTP_AddHeader";SMTP_AddHeader($SMTP_ID;"X-4Ddemo:";◇VERSION))))
   :(Not(VérifErreur("SMTP_Body";SMTP_Body($SMTP_ID;vMessage))))
   :(Not(VérifErreur("SMTP_Send";SMTP_Send($SMTP_ID))))
   Else
      $EnvoyeOK:=True  `Message composé et envoyé avec succès
```

```
End case

If($SMTP_ID#0)  `Si un message a été créé en mémoire, nous devons l'effacer maintenant
   $OK:=VérifErreur("SMTP_Clear";SMTP_Clear($SMTP_ID))
End if
```

Note : Pour plus d'informations sur cet emploi particulier de la structure **Au cas ou**, reportez-vous à l'**Annexe A, Conseils de programmation**.

L'exemple suivant fournit le code de la méthode ***VérifErreur***. Cette méthode reçoit deux paramètres : le nom de la commande (*$Commande*) et la valeur de l'erreur (fournie par l'exécution de la commande dans le paramètre de la méthode). ***VérifErreur*** renvoie une valeur booléenne indiquant si la commande a retourné le code d'erreur 0 (zéro). Si ce n'est pas le cas, la valeur retournée (*$0*) est **Faux**, sinon **Vrai**.

```
C_TEXT(vMsgErreur)
$Commande:=$1
$Erreur:=$2
$Resultat:=True
If($Erreur#0)
   $Resultat:=False
   If(◇VOIRERREURS)  `Booléen pour déterminer s'il faut afficher les messages d'erreur
      vMsgErreur:=IT_ErrorText($Erreur)
      ALERT("ERREUR ---"+Char(13)+"Commande : "+$Commande+Char(13)+"Erreur
Code:"+String($Erreur)+Caractere(13)+"Description : "+vMsgErreur)
   End if
End if
$0:=$Resultat
```

SMTP_Cc

SMTP_Cc (smtp_ID ; copieConforme {; supprimerOption}) -> Résultat

Paramètre	Type		Description
smtp_ID	Entier long	➡	Référence de message
copieConforme	Texte	➡	Adresse électronique ou Liste d'adresses
supprimerOption	Entier	➡	0 = Ajouter, 1 = Remplacer, 2 = Supprimer
Résultat	Entier	↩	Code d'erreur

Description

La commande *SMTP_Cc* ajoute des destinataires en "copie conforme" (Carbon copy) au message spécifié par *smtp_ID*. Pour envoyer un message, il n'est pas obligatoire que l'en-tête "Cc" contienne des adresses. Toutes les adresses listées dans les zones d'en-tête "A" et "Cc" d'un message électronique seront visibles par chaque destinataire.

smtp_ID contient l'identifiant d'un message électronique créé avec la commande *SMTP_New*.

Le paramètre *copieConforme* contient une ou plusieurs adresses électroniques.

Le paramètre *supprimerOption* vous permet de préciser s'il faut conserver ou supprimer la zone d'en-tête "Cc" éventuellement existante :

- Si vous passez 0 (zéro), le contenu du paramètre passé est ajouté au contenu de l'en-tête existant.
- Si vous passez 1, le contenu du paramètre passé remplace le contenu de l'en-tête existant. Dans ce cas, si vous avez passé une chaîne vide dans *copieConforme*, l'en-tête "Cc" est supprimé.
- Si vous passez 2, l'en-tête "Cc" est supprimé du message.
- Si *supprimerOption* est omis, par défaut la valeur 0 est utilisée.

Exemple

Voir l'exemple de la commande *SMTP_Body*.

SMTP_Charset

SMTP_Charset (encoderEntêtes ; jeuCorps) -> Résultat

Paramètre	Type		Description
encoderEntêtes	Entier	➡	-1 = Utiliser le paramétrage courant, 0 = Utiliser la valeur par défaut, 1 = Convertir dans le jeu de caractères spécifié
jeuCorps	Entier	➡	-1 = Utiliser le paramétrage courant, 0 = Utiliser la valeur par défaut, 1 = Convertir dans le jeu de caractères spécifié
Résultat	Entier	↩	Code d'erreur

Description

La commande **SMTP_Charset** automatise le support des caractères étendus dans les messages lors de leur envoi par la commande *SMTP_QuickSend* ou *SMTP_Send*.

La commande *SMTP_Charset* permet, d'une part, d'indiquer si le jeu de caractères défini dans le paramètre *charset&Encodage* de la commande **SMTP_SetPrefs** doit être appliqué aux en-têtes, nom de fichiers joints et corps des messages à envoyer ; d'autre part, elle permet de définir si un en-tête (ou nom de fichier joint) comportant des caractères étendus doit être encodé sous la forme "=?ISO-8859-1?Q?Test=E9?= …", conformément au RFC 1342.
Cette commande a une portée globale et interprocess : elle agit sur tous les messages ultérieurs envoyés à l'aide des commandes **SMTP_QuickSend** et **SMTP_Send** et ce, dans tous les process 4D.

La commande *SMTP_Charset* est particulièrement utile pour le traitement des caractères étendus dans les en-têtes "Subject" ou les noms insérés dans les adresses (par exemple, pour l'encodage d'adresses sous la forme "*=?ISO-8859-1?Q?Test=E9?= <test@n.net>*").

Les en-têtes des messages et les noms des fichiers joints seront encodés de la manière suivante, d'après la RFC 1342 :

- Pour les en-têtes "Subject", "Comments" et les noms des fichiers joints : toute la chaîne est encodée en base64 si elle comporte des caractères étendus ;
- Pour les en-têtes "From", "To", "Cc", "Bcc", "Sender", "ReplyTo", "InReplyTo" :
  - Ce qui est entre < > est systématiquement considéré comme une adresse eMail et n'est jamais encodé ;
  - Les caractères séparateurs tels que SPC < > ( ) @ , ; : \ " / ? . = ne sont jamais encodés ;
  - Ce qui se trouve entre deux séparateurs est encodé en base 64 s'il y a des caractères étendus.
  - Exemples d'adresses :
    someone@somewhere n'est pas encodé ;
    Michèle *<michele@somewhere>*, seul le mot Michèle est encodé.
- Les autres en-têtes ne sont pas encodés.

Le paramètre *encoderEntêtes* définit les traitements à appliquer aux champs d'en-tête lors de l'envoi des messages. Par défaut, ce paramètre a pour valeur 0.

- -1 : Utiliser les paramètrages courants ;
- 0 : Valeur par défaut : jeu de caractères UTF-8 pour "Subject", ISO-8859-1 pour les autres champs ;
- 1 : Le jeu de caractères pour les en-têtes et les noms des fichiers joints est défini par le paramètre *charset&Encodage* de la commande **SMTP_SetPrefs**.

Note : Les en-têtes de type X_Mailer doivent être en ASCII US.

Le paramètre *jeuCorps* définit les traitements à appliquer au corps du message lors de son envoi. Par défaut, ce paramètre a pour valeur 0.

- -1 : Utiliser les paramètrages courants ;
- 0 : Valeur par défaut : jeu de caractères UTF-8 base64
- 1 : Utiliser jeu de caractères et encodage définis par le paramètre *charset&Encodage* de la commande **SMTP_SetPrefs**.

Note : Nous recommandons d'utiliser les paramétrages par défaut, adaptés à la plupart des systèmes/applications.

Exemple

Exemple de traitement des caractères étendus :

```
SMTP_SetPrefs(1;1;0)
$err:=SMTP_Charset(1;1)
$err:=SMTP_QuickSend("monmail.com";"monadresse";"destination";"L'euro €";"Le symbole de l'Euro est €")
 //Le sujet et le corps du message sont convertis en UTF-8,
 //Le sujet est encodé conformément au RFC 1342
```

SMTP_Clear

SMTP_Clear (smtp_ID) -> Résultat

Paramètre	Type		Description
smtp_ID	Entier long	➡	Référence du message
		⬅	0 si exécution correcte
Résultat	Entier	↩	Code d'erreur

Description

La commande *SMTP_Clear* supprime le message désigné par *smtp_ID*, libérant la mémoire utilisée lors de sa création. A chaque appel de *SMTP_New* doit correspondre un appel à *SMTP_Clear*.

smtp_ID contient l'identifiant du message électronique créé avec la commande *SMTP_New*.

Si la commande *SMTP_Clear* a été correctement exécutée, *smtp_ID* retourne la valeur 0 (zéro).

Exemple

Reportez-vous aux exemples des commandes *SMTP_Body* et *SMTP_Send*.

SMTP_Comments

SMTP_Comments (smtp_ID ; msgCommentaires {; supprimerOption}) -> Résultat

Paramètre	Type		Description
smtp_ID	Entier long	➡	Référence de message
msgCommentaires	Texte	➡	Texte du commentaire
supprimerOption	Entier	➡	0 = Remplacer sauf si msgCommentaires vide, 1 = Remplacer, 2 = Supprimer
Résultat	Entier	↩	Code d'erreur

Description

La commande *SMTP_Comments* permet d'ajouter des commentaires au message sans modifier le corps du message. Les commentaires n'apparaissent que dans la zone d'en-tête. De nombreux logiciels de messagerie n'affichent pas l'intégralité des en-têtes du message.

smtp_ID contient l'identifiant d'un message électronique créé avec la commande *SMTP_New*.

Le paramètre *msgCommentaires* contient le texte que vous souhaitez placer en tant que commentaire dans l'en-tête du message.

Attention : Le texte ne doit pas contenir de retours à la ligne (code ASCII=10). Un retour à la ligne désigne la fin de la section d'en-tête et le début du corps du texte. Les en-têtes suivants risquent alors d'être considérés comme le corps du texte et de ne pas être correctement reconnus par le logiciel du serveur ou du client. Pour plus d'informations sur les zones d'en-tête, veuillez consulter la RFC 822.

Le paramètre optionnel *supprimerOption* vous permet de préciser s'il faut conserver ou supprimer l'en-tête "Commentaires" éventuellement existant :

- Si vous passez 0 (zéro), le contenu du paramètre *msgCommentaires* remplace dans la zone d'en-tête tout texte éventuellement présent, sauf si vous passez une chaîne vide dans *msgCommentaires*, auquel cas l'en-tête "Commentaires" existant et son contenu sont conservés.
- Si vous passez 1, le contenu du paramètre *msgCommentaires* remplace dans la zone d'en-tête tout texte éventuellement présent. Dans ce cas, si vous passez une chaîne vide dans *msgCommentaires*, l'en-tête "Commentaires" existant est supprimé.
- Si vous passez 2, l'en-tête "Commentaires" est supprimé du message.
  Si *supprimerOption* est omis, par défaut la valeur 0 est utilisée.

Exemple

Reportez-vous à l'exemple de la commande *SMTP_Body*.

SMTP_Date

SMTP_Date (smtp_ID ; msgDate ; msgHeure ; fuseauHoraire ; décalage {; supprimerOption}) -> Résultat

Paramètre	Type		Description
smtp_ID	Entier long	➡	Référence du message
msgDate	Date	➡	Date de création du message
msgHeure	Heure	➡	Heure de création du message
fuseauHoraire	Entier	➡	Code d'emplacement
décalage	Entier	➡	Dépend de la valeur du paramètre fuseauHoraire
supprimerOption	Entier	➡	0 = Ajouter/Remplacer, 1 = Effacer
Résultat	Entier	↩	Code d'erreur

Description

La commande *SMTP_Date* crée l'en-tête Date du message désigné par le paramètre *smtp_ID*. La date et l'heure fournies à la commande doivent correspondre à l'emplacement courant de la machine envoyant le message. Les paramètres suivants doivent respecter un format spécifique de manière à ce que le serveur de courrier électronique réceptionnant le message puisse déterminer la date et l'heure locales en fonction de la date, l'heure, la zone horaire et le décalage qui lui sont communiqués.

Note : Si un message électronique est composé sans en-tête Date, le serveur SMTP en ajoute une en fonction de ses réglages courants d'heure et de date. Tous les messages électroniques SMTP contiennent un en-tête Date ajouté soit par l'application cliente, soit par le serveur SMTP.

Le paramètre *smtp_ID* contient l'identifiant du message créé avec la commande *SMTP_New*.

Le paramètre *msgDate* est une date 4D qui indique la date de création du message.

Le paramètre *msgHeure* indique l'heure de création du message.

Le paramètre *fuseauHoraire* identifie le fuseau horaire de l'émetteur. Vous pouvez passer toute valeur comprise entre 0 et 6, en fonction des indications suivantes :

- 0 (zéro) permet de spécifier directement dans le paramètre *décalage* le nombre d'heures à soustraire ou à ajouter au temps universel (TU).
- Si vous passez 1, la machine émettrice ajoutera automatiquement le décalage fondé sur la PRAM du Macintosh. Lorsque vous passez 1 dans *fuseauHoraire*, le paramètre *décalage* est inutile. Le fuseau horaire d'un ordinateur Macintosh est déterminé par les préférences système **Date et heure**. Vous devez veiller à l'exactitude de ce paramétrage si les valeurs horaires sont un facteur primordial de votre base de données.
- Les valeurs de 2 à 5 correspondent aux 4 fuseaux horaires des Etats-Unis. Le *décalage* pour chacune de ces valeurs spécifie si cette zone horaire est en heure d'été (*décalage* = 1) ou non (*décalage* = 0).
- Si vous passez 6, vous indiquez que le temps est défini sur 24 heures. Dans ce cas, le décalage est déterminé par le tableau horaire ci-dessous. Passez la valeur de décalage correspondante (de -12 à 12) au code horaire sur 24 heures de l'emplacement de l'émetteur.

La valeur du paramètre *décalage* dépend du code défini dans le paramètre *fuseauHoraire*. Reportez-vous aux descriptions et au tableau horaire suivants pour connaître la valeur correcte à passer dans ce paramètre.

Code	Fuseau horaire	Paramètre de décalage
0	+/- décalage TU	Le décalage est en heures +/-
1	+/- décalage TU	Le décalage n'est pas utilisé, il est fourni par la PRAM du Mac
2	HNE - HAE	(0 = HNE, 1 = HAE)
3	HNC - HAC	(0 = HNC, 1 = HAC)
4	HNR - HAR	(0 = HNR, 1 = HAR)
5	HNP - HAP	(0 = HNP, 1 = HAP)
6	Heure sur 24 heures	Voir tableau ci-dessous

Valeurs de décalage	Codes horaires sur 24 heures
0	Z
-1 à -9	A à I
-10 à -12	K à M
1 à 12	N à Y

Définitions des abréviations

TU	Temps Universel
HNE	Heure normale de l'Est
HAE	Heure avancée de l'Est
HNC	Heure normale du Centre
HAC	Heure avancée du Centre
HNR	Heure normale des Rocheuses
HAR	Heure avancée des Rocheuses
HNP	Heure normale du Pacifique
HAP	Heure avancée du Pacifique

supprimerOption :

- Passez 0 (zéro) dans ce paramètre pour ajouter un nouvel en-tête Date ou remplacer le précédent.
- Passez 1 pour effacer toute valeur préalablement définie dans cet en-tête.
- Si vous omettez ce paramère, la valeur 0 est utilisée.

SMTP_Encrypted

SMTP_Encrypted (smtp_ID ; msgCrypté {; supprimerOption}) -> Résultat

Paramètre	Type		Description
smtp_ID	Entier long	➡	Référence du message
msgCrypté	Texte	➡	Type d'encryptage
supprimerOption	Entier	➡	0 = Remplacer sauf si msgCrypté vide, 1 = Remplacer, 2 = Supprimer
Résultat	Entier	⮌	Code d'erreur

Description

La commande *SMTP_Encrypted* permet d'informer les utilisateurs du type d'encryptage utilisé dans le corps du message. Les commandes Internet de 4D **ne permettent pas** d'encrypter ou de décrypter des messages électroniques. Le cryptage du corps du message relève de la responsabilité du développeur. Si le corps du message est encrypté (avant son affectation via *SMTP_Body*), il est nécessaire d'utiliser cette commande pour indiquer au logiciel de messagerie le type d'encryptage employé.

smtp_ID contient l'identifiant d'un message électronique créé avec la commande *SMTP_New*.

Le paramètre *msgCrypté* indique le type d'encryptage utilisé. L'en-tête "Encryptage" est utilisé par le logiciel de messagerie du destinataire pour décrypter le corps du message reçu. Pour plus amples informations sur le formatage, veuillez consulter la RFC 822.

Attention : Le texte ne doit pas contenir de retours à la ligne (code ASCII=10). Un retour à la ligne désigne la fin de la section d'en-tête et le début du corps du texte. Les en-têtes suivants risquent alors d'être considérés comme le corps du texte et de ne pas être correctement reconnus par le logiciel du serveur ou du client. Pour plus d'informations sur les zones d'en-tête, veuillez consulter la RFC 822.

Le paramètre optionnel *supprimerOption* vous permet de préciser s'il faut conserver ou supprimer l'en-tête "Encryptage" éventuellement existant :

- Si vous passez 0 (zéro), le contenu du paramètre *msgCrypté* remplace dans la zone d'en-tête tout texte éventuellement présent, sauf si vous passez une chaîne vide dans *msgCrypté*, auquel cas l'en-tête "Encryptage" existant et son contenu sont conservés.
- Si vous passez 1, le contenu du paramètre *msgCrypté* remplace dans la zone d'en-tête tout texte éventuellement présent. Dans ce cas, si vous passez une chaîne vide dans *msgCrypté*, l'en-tête "Encryptage" existant est supprimé.
- Si vous passez 2, l'en-tête "Encryptage" est supprimé du message.
  Si *supprimerOption* est omis, par défaut la valeur 0 est utilisée.

SMTP_From

SMTP_From (smtp_ID ; msgDe {; supprimerOption}) -> Résultat

Paramètre	Type		Description
smtp_ID	Entier long	➡	Référence du message
msgDe	Texte	➡	Adresse électronique ou Liste d'adresses
supprimerOption	Entier	➡	0 = Ajouter, 1 = Remplacer, 2 = Supprimer
Résultat	Entier	↩	Code d'erreur

Description

La commande *SMTP_From* contient la ou les adresse(s) électronique(s) de la ou des personne(s) devant figurer dans l'en-tête "Emetteur(s)" (From) du message. Ces personnes sont responsables de la création ou de l'envoi du message. Généralement, l'en-tête "Emetteur(s)" contient l'adresse de la personne qui a composé et envoyé le message. Cependant, lorsque le message est créé par un groupe de personnes, chaque membre doit être individuellement listé dans l'en-tête "Emetteur(s)".

L'en-tête "Emetteur(s)" est obligatoire. Si une adresse figure dans l'en-tête "Emetteur(s)", la présence de l'en-tête "Expéditeur" (Sender) est optionnelle.

smtp_ID contient l'identifiant du message électronique créé avec la commande *SMTP_New*.

msgDe peut contenir une ou plusieurs adresses électroniques. Toutes les adresses figurant dans l'en-tête "Emetteur(s)" sont visibles par les destinataires du message.

Note sur les réponses automatiques : En l'absence d'en-tête "Réponse à" (ReplyTo) dans le message identifié par *smtp_ID*, toutes les réponses au message seront adressées à chaque personne figurant dans l'en-tête "Emetteur(s)".

Le paramètre *supprimerOption* vous permet de préciser s'il faut conserver ou supprimer la zone d'en-tête "Emetteur(s)" éventuellement existante :

- Si vous passez 0 (zéro), le contenu du paramètre passé est ajouté au contenu de l'en-tête existant.
- Si vous passez 1, le contenu du paramètre passé remplace le contenu de l'en-tête existant. Dans ce cas, si vous avez passé une chaîne vide dans *msgDe*, l'en-tête "Emetteur(s)" est supprimé.
- Si vous passez 2, l'en-tête "Emetteur(s)" est supprimé du message.
- Si *supprimerOption* est omis, par défaut la valeur 0 est utilisée.

Exemple

Dans cet exemple, trois personnes composent un message, concernant une modification de politique de l'entreprise, qui doit être distribué à tout le personnel de la société. Les réponses à ce message seront adressées aux personnes présentes dans l'en-tête "Emetteur(s)".

```
$Emetteur:="prez@acme.com,vp@acme.com,cfo@acme.com"
$Erreur:=SMTP_From($smtp_id;$Emetteur;0)
$Erreur:=SMTP_Subject($smtp_id;"Changement de politique de l'entreprise";0)
$Erreur:=SMTP_To($smtp_id;◇Tous_employés;0)
```

SMTP_GetPrefs

SMTP_GetPrefs (retoursLigne ; charset&Encodage ; longueurLigne) -> Résultat

Paramètre	Type		Description
retoursLigne	Entier	⇐	0 = Ne pas ajouter, 1 = Ajouter RetoursLigne
charset&Encodage	Entier long	⇐	Jeu de caractères du corps, en-têtes, noms des fichiers joints et encodage du corps
longueurLigne	Entier long	⇐	Longueur de ligne maximale
Résultat	Entier	⇒	Code d'erreur

Description

La commande **SMTP_GetPrefs** renvoie les paramètres courants des préférences SMTP. Les valeurs par défaut sont retournées, à moins qu'un appel préalable à *SMTP_SetPrefs* ne les ait modifiées. Pour une description complète de ces paramètres, reportez-vous à la commande *SMTP_SetPrefs*.

Le paramètre *retoursLigne* retourne la manière dont les commandes SMTP gèrent les retours chariot dans le corps d'un message.

Le paramètre *charset&Encodage* retourne les paramétrages courants pour le corps, les en-têtes et les noms de fichiers joints. Reportez-vous au tableau *charset&Encodage* de la commande *SMTP_SetPrefs* pour la description des combinaisons indiquées par chaque valeur.

Le paramètre *longueurLigne* retourne la longueur maximale courante de ligne du texte dans le corps du message.

SMTP_Host

SMTP_Host (smtp_ID ; nomServeur {; supprimerOption}) -> Résultat

Paramètre	Type		Description
smtp_ID	Entier long	➡	Référence du message
nomServeur	Chaîne	➡	Nom ou adresse IP du serveur
supprimerOption	Entier	➡	0 = Utiliser ou remplacer, 1 = Supprimer
Résultat	Entier	↩	Code d'erreur

Description

Tous les messages créés et envoyés avec les commandes SMTP doivent être dirigés vers un serveur SMTP spécifique. 4D Internet Commands ne distribue pas de courrier directement à chaque destinataire, mais uniquement au serveur SMTP désigné par cette commande. Le serveur SMTP est chargé de la résolution des éventuelles erreurs d'adresse et de la distribution du message.

smtp_ID est l'identifiant du message électronique créé avec la commande *SMTP_New*.

nomServeur contient le nom ou l'adresse IP du serveur SMTP qui gèrera la distribution du message.

supprimerOption vous permet de supprimer ou non la valeur courante du paramètre *nomServeur* :

- Si vous passez 0 (zéro), le serveur utilisé est celui spécifié par *nomServeur*.
- Si vous passez 1, la valeur de *nomServeur* définie pour le message identifié par *smtp_ID* est supprimée.
- Si vous omettez ce paramètre, par défaut la valeur 0 est utilisée.

Exemple

Reportez-vous aux exemples des commandes *SMTP_Body* et *SMTP_Send*.

SMTP_InReplyTo

SMTP_InReplyTo (smtp_ID ; msgEnRéponseA {; supprimerOption}) -> Résultat

Paramètre	Type		Description
smtp_ID	Entier long	⇒	Référence du message
msgEnRéponseA	Texte	⇒	Texte en réponse à
supprimerOption	Entier	⇒	0 = Remplacer sauf si msgEnReponseA vide, 1 = Remplacer, 2 = Supprimer
Résultat	Entier	⇐	Code d'erreur

Description

La commande *SMTP_InReplyTo* insère, lors de la réponse à un message, le texte du message d'origine.

smtp_ID contient l'identifiant d'un message électronique créé avec la commande *SMTP_New*.

Le paramètre *msgEnRéponseA* contient le texte des messages précédents auxquels se rapporte ce message. Pour toute référence se rapportant à son formatage, veuillez consulter la RFC 822.

Attention : Le texte ne doit pas contenir de retours à la ligne (code ASCII=10). Un retour à la ligne désigne la fin de la section d'en-tête et le début du corps du texte. Les en-têtes suivants risquent alors d'être considérés comme le corps du texte et de ne pas être correctement reconnus par le logiciel du serveur ou du client. Pour plus d'informations sur les zones d'en-tête, veuillez consulter la RFC 822.

Le paramètre optionnel *supprimerOption* vous permet de préciser s'il faut conserver ou supprimer l'en-tête "Réponse A" éventuellement existant :

- Si vous passez 0 (zéro), le contenu du paramètre *msgEnRéponseA* remplace dans la zone d'en-tête tout texte éventuellement présent, sauf si vous passez une chaîne vide dans *msgEnRéponseA*, auquel cas l'en-tête "Réponse A" existant et son contenu sont conservés.
- Si vous passez 1, le contenu du paramètre *msgEnRéponseA* remplace dans la zone d'en-tête tout texte éventuellement présent. Dans ce cas, si vous passez une chaîne vide dans *msgEnRéponseA*, l'en-tête "Réponse A" existant est supprimé.
- Si vous passez 2, l'en-tête "Réponse A" est supprimé du message.
  Si *supprimerOption* est omis, la valeur 0 est utilisée par défaut.

SMTP_Keywords

SMTP_Keywords (smtp_ID ; msgMotsClés {; supprimerOption}) -> Résultat

Paramètre	Type		Description
smtp_ID	Entier long	⇒	Référence de message
msgMotsClés	Texte	⇒	Liste de mots-clés
supprimerOption	Entier	⇒	0 = Remplacer sauf si msgMotsClés vide, 1 = Remplacer, 2 = Supprimer
Résultat	Entier	⇐	Code d'erreur

Description

Le commande *SMTP_Keywords* permet d'insérer des mots-clés dans l'en-tête "Mots-clés" (Keywords) du message désigné par *smtp_ID*.

smtp_ID est l'identifiant d'un message électronique créé avec la commande *SMTP_New*.

Le paramètre *msgMotsClés* contient un ou plusieurs mots-clés, séparés par des virgules. Pour plus d'informations sur le formatage de ce paramètre, veuillez consulter la RFC 822.

Attention : Le texte ne doit pas contenir de retours à la ligne (code ASCII=10). Un retour à la ligne désigne la fin de la section d'en-tête et le début du corps du texte. Les en-têtes suivants risquent alors d'être considérés comme le corps du texte et de ne pas être correctement reconnus par le logiciel du serveur ou du client. Pour plus d'informations sur les zones d'en-tête, veuillez consulter la RFC 822.

Le paramètre optionnel *supprimerOption* vous permet de préciser s'il faut conserver ou supprimer l'en-tête "Mots-clés" éventuellement existant :

- Si vous passez 0 (zéro), le contenu du paramètre *msgMotsClés* remplace dans la zone d'en-tête tout texte éventuellement présent, sauf si vous passez une chaîne vide dans *msgMotsClés*, auquel cas l'en-tête "Mots-clés" existant et son contenu sont conservés.
- Si vous passez 1, le contenu du paramètre *msgMotsClés* remplace dans la zone d'en-tête tout texte éventuellement présent. Dans ce cas, si vous passez une chaîne vide dans *msgMotsClés*, l'en-tête "Mots-clés" existant est supprimé.
- Si vous passez 2, l'en-tête "Mots-clés" est supprimé du message.
  Si *supprimerOption* est omis, par défaut la valeur 0 est utilisée.

SMTP_MessageID

SMTP_MessageID (smtp_ID ; message_ID {; option}) -> Résultat

Paramètre	Type		Description
smtp_ID	Entier long	⇒	Référence de message
message_ID	Texte	⇒	Identifiant unique du message
option	Entier	⇒	0 = Ajouter (défaut), 1 = Remplacer, 2 = Supprimer
Résultat	Entier	⇐	Code d'erreur

Description

La commande **SMTP_MessageID** permet d'ajouter un champ "message-id" dans l'en-tête du message dont la référence est *smtp_ID*. Cet identifiant unique est utilisé notamment sur les forums ou listes de messagerie publiques. En général, les serveurs de messagerie ajoutent automatiquement cet en-tête aux messages qu'ils émettent. Cette commande vous permet de définir son contenu.

smtp_ID contient l'identifiant d'un message électronique créé avec la commande **SMTP_New**.

Passez dans *message_ID* l'identifiant à associer au message. Le contenu à passer est en principe libre, toutefois par convention il sera généralement de la forme "*lettresOuChiffres@nomdomaine*", par exemple "abcdef.123456@4d.com". A noter que certains serveurs de messagerie (par exemple Gmail) ne reconnaissent pas les en-têtes "message-id" personnalisés et les remplacent s'ils ne sont pas sous cette forme.

Le paramètre *option* vous permet de préciser s'il faut conserver ou supprimer l'en-tête *message_ID* éventuellement existant :

- Si vous passez 0 (valeur par défaut si le paramètre est omis), le contenu du paramètre passé est ajouté au contenu existant.
- Si vous passez 1, le contenu du paramètre passé remplace le contenu existant.
- Si vous passez 2, le contenu existant est supprimé du message.

Exemple

Dans cet exemple, un message avec en-tête "message-id" spécifique est envoyé pour chaque enregistrement de la table [Admins] :

```
$erreur:=SMTP_New($smtp_id)
$erreur:=SMTP_Host($smtp_id;"infoserv.com")
$erreur:=SMTP_From($smtp_id;"info@infoserv.com")
$erreur:=SMTP_Subject($smtp_id;"Statistiques générales")
FIRST RECORD([Admins])
For($i;1;Records in selection([Admins]))
   $erreur:=SMTP_Body($smtp_id;$Stats)
   $erreur:=SMTP_To($smtp_id;[Admins]Email;1)  //Remplacer l'en-tête "A" par une nouvelle valeur
   $erreur:=SMTP_MessageID($smtp_id;[Admins]ID+"@infoserv.com";1)  //Utilisation de l'id de l'admin
   $erreur:=SMTP_Send($smtp_id)
   NEXT RECORD([Admins])
End for
$erreur:=SMTP_Clear($smtp_id)
```

SMTP_New

SMTP_New (smtp_ID) -> Résultat

Paramètre	Type		Description
smtp_ID	Entier long	⇐	Référence du nouveau message
Résultat	Entier	⇒	Code d'erreur

Description

La commande *SMTP_New* doit être la première commande exécutée dans la séquence de création d'un message électronique SMTP, sauf si *SMTP_QuickSend* est utilisée. *SMTP_New* crée un nouveau message en mémoire et renvoie sa référence sous forme d'entier long dans le paramètre *smtp_ID*. Les commandes SMTP ultérieures devront utiliser cette référence pour remplir les informations d'en-tête et de corps du message avant que *SMTP_Send* soit appelée.

Chaque appel de *SMTP_New* doit avoir un appel correspondant de *SMTP_Clear*. Après avoir envoyé un message, l'appel de *SMTP_Clear* libère la mémoire occupée par le contenu du message.

Le paramètre *smtp_ID* retourne le numéro d'identification du message créé. Cet ID est utilisé pour toutes les références ultérieures à ce message. Il est possible d'ouvrir simultanément plusieurs nouveaux messages, le *smtp_ID* retourné pour chacun d'eux fournit un moyen d'identifier le message auquel doit s'appliquer une commande.

Exemples

Reportez-vous aux exemples des commandes *SMTP_Body* et *SMTP_Send*.

SMTP_QuickSend

SMTP_QuickSend (nomServeur ; msgDe ; msgA ; msgObjet ; message {; paramSession}{; port}{; nomUtilisateur ; motDePasse}) -> Résultat

Paramètre	Type		Description
nomServeur	Chaîne	➡	Nom ou adresse IP du serveur
msgDe	Texte	➡	Adresse électronique ou Liste d'adresses
msgA	Texte	➡	Adresse électronique ou Liste d'adresses
msgObjet	Texte	➡	Objet du message (UTF-8 par défaut)
message	Texte	➡	Message (UTF-8 par défaut)
paramSession	Entier long	➡	0 ou omis = Ne pas utiliser SSL mais bascule permise, 1 = Utiliser SSL, 2 = Ne jamais utiliser SSL (bascule non permise), 4 = Envoyer texte HTML sans SSL, 5 = Envoyer texte HTML avec SSL, 8 = Envoyer Mime HTML sans SSL/TLS, 9 = Envoyer Mime HTML avec SSL/TLS
port	Entier long	➡	Numéro de port à utiliser
nomUtilisateur	Texte	➡	Nom d'utilisateur pour l'authentification
motDePasse	Texte	➡	Mot de passe pour l'authentification
Résultat	Entier	↩	Code d'erreur

Description

La commande **SMTP_QuickSend** vous permet de créer et d'envoyer un message en une seule commande. Si vous avez besoin d'un plus grand contrôle sur votre message, ou s'il est plus complexe, utilisez plutôt la commande **SMTP_New**.

Note : Cette commande ne peut être utilisée dans les bases de données converties fonctionnant en mode "non-unicode".

Le paramètre *nomServeur* contient le nom ou l'adresse IP du serveur SMTP auquel le message sera envoyé pour distribution.

Le paramètre *msgDe* contient une ou plusieurs adresses électroniques complètes indiquant l'expéditeur initial du message. Toutes les adresses figurant dans l'en-tête Emetteur (From) sont visibles par tous les destinataires du message.

Le paramètre *msgA* contient une ou plusieurs adresses électroniques complètes. Les adresses figurant dans l'en-tête *msgA* recevront chacune une copie originale du message. Chaque destinataire du message pourra visualiser les autres adresses électroniques auxquelles le message a été envoyé.

Le paramètre *objet* contient un texte concis décrivant l'objet du message. Le paramètre *message* contient le corps du message électronique.

Note : Par défaut, l'objet et le corps du message sont encodés en UTF-8, ce qui garantit que les caractères envoyés seront correctement interprétés par la quasi totalité des clients de messagerie. Si vous souhaitez utiliser un jeu de caractères spécifique, reportez-vous aux commandes **SMTP_SetPrefs** et **SMTP_Charset**.

Le paramètre optionnel *paramSession* vous permet de définir le format du message (texte standard, HTML ou Mime HTML) et le mode d'activation du protocole SSL pour la connexion :

- si vous passez 0 ou omettez ce paramètre, le message sera formaté en texte et envoyé en mode standard non sécurisé. Si le serveur propose une mise à jour en SSL/TLS après l'authentification, la bascule est effectuée automatiquement (fonctionnement du SSL/TLS en mode explicite).
- si vous passez 1, le message sera formaté en texte et envoyé en SSL (mode synchrone),
- si vous passez 2, le message sera formaté en texte et envoyé en mode standard mais sans prise en charge de la mise à jour en SSL/TLS,
- si vous passez 4, le message sera formaté en HTML et envoyé en mode standard,
- si vous passez 5, le message sera formaté en HTML et envoyé en mode SSL/TLS,
- si vous passez 8, le message sera formaté en Mime HTML et envoyé en mode standard,
- si vous passez 9, le message sera formaté en Mime HTML et envoyé en mode SSL/TLS.
  Note : Le Mime HTML (extension de fichier .mht ou .mhtml) est un format d'archive de page Web qui peut combiner du code HTML et des ressources externes telles que des images dans un seul document. Il est pris en charge par de nombreux navigateurs Web ainsi que par MS Word, par exemple. Comme ce format est également pris en charge par les zones 4D Write Pro, vous pouvez facilement enregistrer et envoyer des documents 4D Write Pro par mail avec toutes leurs ressources.

A noter que ces valeurs correspondent à des combinaisons usuelles, le paramètre *paramSession* étant en fait un "champ de bits" (*bit field)*, il permet de définir toute combinaison personnalisée à l'aide des

Opérateurs sur les bits :

Numéro de bit	Format utilisé si le bit est à 1
0	Utiliser SSL ou le paramétrage par défaut, connexion en clair et mise à niveau en SSL si possible.
1	Ne jamais mettre à niveau, rester en mode non encrypté même si SSL possible. Ce bit est ignoré si SSL (bit 0) est sélectionné.
2	Body du message en HTML, définir l'en-tête de façon appropriée.
3	Message MHTML, le bit 2 (HTML) est ignoré. L'utilisateur doit définir tous les en-têtes sauf "A", "De", "Date", et "Objet"

Le paramètre optionnel *port* vous permet de spécifier le numéro de port SMTP à utiliser pour la connexion avec le serveur. Les valeurs les plus fréquemment utilisées sont :

- 25 = port SMTP standard non sécurisé (port par défaut si le paramètre est omis)
- 465 = port SMTPS (SSL/TLS)
- 587 = port SMTP standard mais sécurisé ; passez ce port pour les connexions avec un serveur MS Exchange (mode explicite).

Les paramètres optionnels *nomUtilisateur* et *motDePasse* permettent d'authentifier l'émetteur auprès du serveur de messagerie. Ces paramètres doivent être passés ensemble. A noter que le mode d'authentification le plus sécurisé pris en charge par le serveur sera utilisé (à l'image du mode par défaut de la commande **SMTP_Auth**).

Exemple 1

Exemple d'utilisation de cette commande :

```
$NomServeur:="www.4d.com"
$MsgÀ:="adupont@4d.fr"
$MsgDe:="jsmith@4d.com"
$Objet:="Rapport de ventes"
$Message:="Pouvez-vous m'envoyer le rapport des ventes de janvier 2009 ? Merci."
$Erreur:=SMTP_QuickSend($NomServeur;$MsgDe;$MsgÀ;$Objet;$Message;1)
If($Erreur#0)
   ALERT("Erreur: SMTP_QuickSend"+Char(13)+IT_ErrorText($Erreur))
End if
```

Exemple 2

Exemple d'utilisation de la commande pour un envoi de message sécurisé via un serveur MS Exchange:

```
$NomServeur:="exchange.4d.com"
$MsgA:="adupont@gmail.com"
$MsgDe:="un.utilisateur@4d.com"
$Objet:="Message de test"
$Message:="Ceci est un test d'envoi en mode sécurisé. Merci de ne pas répondre."
$Erreur:=SMTP_QuickSend($NomServeur;$MsgDe;$MsgA;$Objet;$Message;0;587;"un.utilisateur";"@!motdepasse@!")
```

Exemple 3

Envoi d'un message en HTML avec SSL/TLS :

```
$Host:="smtp.gmail.com"
$ToAddress:="john@4d.com"
$FromAddress:="jeanmarc@gmail.com"
$Subject:="Message HTML"
$Message:="Rendez-vous au <b>café du monde</b> !!"
$Param:=5  //HTML avec SSL
$Port:=465  //Port SSL de gmail
```

```
$User:="jeanmarc@gmail.com"
$Password:="xyz&@!&@"
$Error:=SMTP_QuickSend($Host;$FromAddress;$ToAddress;$Subject;$Message;$Param;$Port;$User;$Password)
```

Exemple 4

Vous avez enregistré un document .mht sur votre disque et souhaitez l'envoyer par email. Pour cela, vous pouvez écrire :

```
$Message:=Document to text("c:\\documents\\invitation.mht")
$Host:="smtp.gmail.com"
$ToAddress:="john@4d.com"
$FromAddress:="harry@gmail.com"
$Subject:="Faisons la fête"
$Param:=9  //MHTML avec SSL
$Port:=465  //Port SSL de gmail
$User:="harry@gmail.com"
$Password:="xyz&@!&@"
$Error:=SMTP_QuickSend($Host;$FromAddress;$ToAddress;$Subject;$Message;$Param;$Port;$User;$Password)
```

SMTP_References

SMTP_References (smtp_ID ; msgRéférences {; supprimerOption}) -> Résultat

Paramètre	Type		Description
smtp_ID	Entier long	➡	Référence de message
msgRéférences	Texte	➡	Texte de référence
supprimerOption	Entier	➡	0 = Remplacer sauf si msgRéférences vide, 1 = Remplacer, 2 = Supprimer
Résultat	Entier	⮌	Code d'erreur

Description

La commande *SMTP_References* insère, lors de la réponse à un message, des références supplémentaires au texte d'origine.

smtp_ID contient l'identifiant d'un message électronique créé avec la commande *SMTP_New*.

Le paramètre *msgRéférences* contient le texte des références supplémentaires. Pour des informations se rapportant à son formatage, veuillez consulter la RFC 822.

Attention : Le texte ne doit pas contenir de retours à la ligne (code ASCII=10). Un retour à la ligne désigne la fin de la section d'en-tête et le début du corps du texte. Les en-têtes suivants risquent alors d'être considérés comme le corps du texte et de ne pas être correctement reconnus par le logiciel du serveur ou du client. Pour plus d'informations sur les zones d'en-tête, veuillez consulter la RFC 822.

Le paramètre optionnel *supprimerOption* vous permet de préciser s'il faut conserver ou supprimer l'en-tête "Références" éventuellement existant :

- Si vous passez 0 (zéro), le contenu du paramètre *msgRéférences* remplace dans la zone d'en-tête tout texte éventuellement présent, sauf si vous passez une chaîne vide dans *msgRéférences*, auquel cas l'en-tête "Références" existant et son contenu sont conservés.
- Si vous passez 1, le contenu du paramètre *msgRéférences* remplace dans la zone d'en-tête tout texte éventuellement présent. Dans ce cas, si vous passez une chaîne vide dans *msgRéférences*, l'en-tête "Références" existant est supprimé.
- Si vous passez 2, l'en-tête "Références" est supprimé du message.
  Si *supprimerOption* est omis, par défaut la valeur 0 est utilisée.

SMTP_ReplyTo

SMTP_ReplyTo (smtp_ID ; réponseA {; supprimerOption}) -> Résultat

Paramètre	Type		Description
smtp_ID	Entier long	➡	Référence du message
réponseA	Texte	➡	Adresse électronique ou Liste d'adresses
supprimerOption	Entier	➡	0 = Ajouter à la liste existante, 1 = Remplacer les anciennes valeurs par les nouvelles, 2 = Supprimer les adresses spécifiées
Résultat	Entier	↩	Code d'erreur

Description

La commande *SMTP_ReplyTo* vous permet de prédéfinir le contenu par défaut de la liste d'adresses qui sera utilisée lorsque le récepteur répondra à un message. Normalement, c'est le contenu de l'en-tête "Emetteur(s)" (From) qui est utilisé dans ce cas. Toutefois, cet en-tête est ignoré lorsqu'un en-tête "Réponse à" (ReplyTo) a été défini pour le message.

Pour le développeur de bases de données, *SMTP_ReplyTo* peut être un outil puissant de contrôle des destinataires des réponses dans des messageries automatiques. La commande permet d'envoyer des réponses à des adresses autres que celles inscrites dans les en-têtes "Emetteur(s)" ou "Expéditeur", par exemple à un compte spécifique créé pour assurer le suivi des réponses.

smtp_ID contient l'identifiant du message électronique créé avec la commande *SMTP_New*.

Le paramètre *réponseA* contient une ou plusieurs adresses électroniques. Les adresses listées dans cet en-tête seront utilisées par le logiciel de messagerie des destinataires comme adresse(s) de réponse par défaut.

Le paramètre *supprimerOption* vous permet de préciser comment gérer la ou les adresse(s) insérée(s) dans *réponseA*.

- Si vous passez 0 (zéro), les nouvelles valeurs seront ajoutées à celles éventuellement présentes dans cet en-tête.
- Si vous passez 1, les valeurs éventuellement présentes sont remplacées par les nouvelles valeurs. Si vous avez passé une chaîne vide dans *réponseA*, vous obtiendrez le même résultat qu'en passant 2.
- Si vous passez 2, l'en-tête ainsi que son contenu sont supprimés.
- Si *supprimerOption* est omis, la valeur 0 est utilisée par défaut.

Exemple

Dans cet exemple, trois cadres composent un message concernant un changement de politique de l'entreprise. Ce message doit être adressé à chaque employé de la société par la secrétaire. Les réponses à ce message seront retournées à la secrétaire et au "service_du_personnel", et non aux cadres.

```
$Emetteur:="prez@acme.com,vp@acme.com,cfo@acme.com"
$Erreur:=SMTP_From($smtp_id;$Emetteur;0)
$Erreur:=SMTP_Sender($smtp_id;"secretaire@acme.com";0)
$Erreur:=SMTP_ReplyTo($smtp_id;"secretaire@acme.com,service_du_personnel@acme.com";0)
$Erreur:=SMTP_Subject($smtp_id;"Changement de politique de l'entreprise";0)
$Erreur:=SMTP_To($smtp_id;◇Tous_employés;0)
```

SMTP_Send

SMTP_Send (smtp_ID {; paramSession}) -> Résultat

Paramètre	Type		Description
smtp_ID	Entier long	➡	Référence du message
paramSession	Entier long	➡	0 ou omis = Ne pas utiliser SSL mais bascule permise, 1 = Utiliser SSL, 2 = Ne jamais utiliser SSL (bascule non permise)
Résultat	Entier	↩	Code d'erreur

Description

La commande *SMTP_Send* envoie le message référencé par *smtp_ID,* mais n'efface pas les données de la mémoire.

smtp_ID est l'identifiant du message électronique créé avec la commande *SMTP_New*.

Le paramètre optionnel *paramSession* vous permet de définir le mode d'activation du protocole SSL pour la connexion :

- si vous passez 0 ou omettez ce paramètre, le message sera envoyé en mode standard non sécurisé. Si le serveur propose une mise à jour en SSL/TLS après authentification, la bascule est effectuée automatiquement (fonctionnement du SSL/TLS en mode explicite).
- si vous passez 1, l'envoi du message sera effectué en SSL (mode synchrone),
- si vous passez 2, l'envoi du message sera effectué en mode standard mais sans prise en charge de la mise à jour en SSL/TLS.

Notes concernant l'utilisation de STARTTLS (mode explicite)

A compter de la version 13.2, 4D Internet Commands prend en charge les connexions STARTTLS en mode explicite. Le principe est que la connexion s'effectue en mode standard puis est "mise à niveau" en SSL/TLS après la phase d'authentification. Reportez-vous à l'exemple 2 pour une illustration de ce mécanisme.

- La connexion initiale doit être effectuée sur un port non-SSL/TLS mais qui n'est pas le port par défaut (25). Vous devez donc appeler la commande **IT_SetPort** avant **SMTP_Send** afin de désigner le port utilisé pour la connexion SMTP initiale. Pour une connexion à un serveur MS Exchange, il faut utiliser le port 587.
- La connexion doit être authentifiée, vous devez donc appeler la commande **SMTP_Auth**. Seul le mode d'authentification LOGIN est pris en charge par 4D Internet Commands pour communiquer avec un serveur MS Exchange. Vous pouvez soit passer ce mode, soit laisser le mode par défaut (dans ce cas le mode le plus sécurisé du serveur est utilisé) :

```
$erreur:=SMTP_Auth($smtp_id;"user.name";"password";2)  // OK mode LOGIN
$erreur:=SMTP_Auth($smtp_id;"user.name";"password")  // OK mode LOGIN défini par le serveur
```

Exemple 1

Dans cet exemple, un message est créé et les éléments statiques sont définis. Ensuite, pour chaque enregistrement de la table [Personnes], le message est personnalisé et envoyé.

```
$erreur:=SMTP_New($smtp_id)
$erreur:=SMTP_Host($smtp_id;"wkrp.com")
$erreur:=SMTP_From($smtp_id;"herb_tarlick@wkrp.com")
$erreur:=SMTP_ReplyTo($smtp_id;"bigguy@wkrp.com")
$erreur:=SMTP_Subject($smtp_id;"Promotions sur les espaces publicitaires !")
FIRST RECORD([Personnes])
For($i;1;Records in selection([Personnes]))
   If([Personnes]VentesACeJour>100000)
      $Corps:=◇GrdTexteDisque
   Else
      $Corps:=◇PttTexteDisque
```

```
      Fin de Si
      $Corps:=Replace string($TexteConstant;"<Salutations>";[Personnes]Prénom)
      $erreur:=SMTP_To($smtp_id;[Personnes]Email;1) `Remplacer l'en-tête "A" par une nouvelle valeur
      $erreur:=SMTP_Body($smtp_id;$Corps)
      $erreur:=SMTP_Send($smtp_id)
      NEXT RECORD([Personnes])
   End for
   $erreur:=SMTP_Clear($smtp_id)
```

Exemple 2

Cet exemple envoie un message de test via un serveur Exchange en STARTTLS :

```
$erreur:=SMTP_New($smtp_id)
$erreur:=SMTP_Host($smtp_id;"exchange.4d.com")
$erreur:=SMTP_From($smtp_id;"username@4d.com")
$erreur:=SMTP_ReplyTo($smtp_id;"username@4d.com")
$erreur:=SMTP_Subject($smtp_id;"Test de message")
$erreur:=SMTP_Auth($smtp_id;"username";"!%@password")   //utiliser des identifiants valides
$Body:="Ceci est un test d'envoi de message via Exchange, merci de ne pas répondre"
$erreur:=IT_SetPort(2;587)  //mode SMTP standard, port 587 pour Exchange
$erreur:=SMTP_To($smtp_id;"destinataire@gmail.com")
$erreur:=SMTP_Body($smtp_id;$Body)
$erreur:=SMTP_Send($smtp_id;0)  //Envoi en mode 'upgradable'
ALERT(String($erreur));
```

SMTP_Sender

SMTP_Sender (smtp_ID ; msgExpéditeur {; supprimerOption}) -> Résultat

Paramètre	Type		Description
smtp_ID	Entier long	➡	Référence du message
msgExpéditeur	Texte	➡	Adresse électronique (1 seulement)
supprimerOption	Entier	➡	0 = Ajouter, 1 = Remplacer, 2 = Supprimer
Résultat	Entier	↩	Code d'erreur

Description

La commande *SMTP_Sender* permet de paramétrer l'adresse e-mail de la personne qui envoie le message. Cette commande est destinée à être utilisée lorsque l'expéditeur n'est pas le véritable auteur du message, ou pour désigner la personne qui, dans un groupe d'auteurs, a réellement envoyé le message. Cette commande n'est pas nécessaire lorsque le contenu du champ "Expéditeur" (Sender) est redondant avec celui du champ "Emetteur(s)" (From).

Lorsque les messages sont créés et envoyés automatiquement par un programme, la zone d'en-tête "Expéditeur" doit désigner le compte e-mail de l'administrateur du programme et non le compte géré par le programme.

smtp_ID contient l'identifiant du message électronique créé avec la commande *SMTP_New*.

Le paramètre *msgExpéditeur* contient l'adresse électronique à insérer dans l'en-tête "Expéditeur" du message. **Une** seule adresse peut être spécifiée pour cet en-tête.

Le paramètre *supprimerOption* permet de spécifier s'il faut conserver ou supprimer la zone d'en-tête "Expéditeur" éventuellement existante :

- Si vous passez 0 (zéro), le contenu du paramètre passé est ajouté au contenu de l'en-tête existant.
- Si vous passez 1, le contenu du paramètre passé remplace le contenu de l'en-tête existant. Dans ce cas, si vous avez passé une chaîne vide, l'en-tête "Expéditeur" est supprimé.
- Si vous passez 2, l'en-tête "Expéditeur" est supprimé du message.
- Si ce paramètre est omis, par défaut la valeur 0 est utilisée.

Exemple

Dans cet exemple, trois cadres composent un message concernant une modification de politique de l'entreprise. Ce message doit être distribué à tout le personnel de la société par la secrétaire. Les réponses éventuelles devront être envoyées aux trois personnes répertoriées dans l'en-tête "Emetteur(s)".

```
$Emetteur:="prez@acme.com,vp@acme.com,cfo@acme.com"
$Erreur:=SMTP_From($smtp_id;$Emetteur;0)
$Erreur:=SMTP_Sender($smtp_id;"secretaire@acme.com";0)
$Erreur:=SMTP_Subject($smtp_id;"Changement de politique de l'entreprise";0)
$Erreur:=SMTP_To($smtp_id;◇Tous_employés;0)
```

SMTP_SetPrefs

SMTP_SetPrefs (retoursLigne ; charset&Encodage ; longueurLigne) -> Résultat

Paramètre	Type		Description
retoursLigne	Entier	➡	1 = [défaut] Ajouter, 0 = Ne pas ajouter, -1 = Aucune modification
charset&Encodage	Entier long	➡	Jeu de caractères du corps, en-têtes, noms des fichiers joints et encodage du corps (-1 = pas de changement)
longueurLigne	Entier long	➡	Longueur de ligne maximale (0 = [défaut] Détection auto, -1 = Aucune modification)
Résultat	Entier	↩	Code d'erreur

Description

La commande **SMTP_SetPrefs** définit les préférences des messages SMTP à envoyer. Elle a une portée globale et interprocess : elle agit sur tous les messages ultérieurs créés avec des commandes SMTP et ce, dans tous les process 4D. Les options paramétrables s'appliquent au message lors de son envoi au serveur SMTP à l'aide des commandes *SMTP_QuickSend* ou *SMTP_Send*.

A la différence des applications Mac qui considèrent un retour chariot seul comme étant un marqueur de fin de ligne ou de paragraphe, les serveurs SMTP requièrent la combinaison de caractères retour chariot/retour à la ligne (CR/LF) pour indiquer la fin d'une ligne.

Le paramètre *retoursLigne* spécifie comment gérer les retours chariot dans le corps d'un message :

- Si vous passez 0 (zéro), le texte du corps du message est inchangé, ce qui vous permet de gérer vos propres ajouts de retours à la ligne.
- Si vous passez 1 (paramètre par défaut), toutes les combinaisons retour chariot/retour ligne seront automatiquement remplacées par des retours chariot seuls.
- Si vous passez -1, la valeur courante du paramètre est inchangée.
  En cas de doute, choisissez 1, la valeur par défaut.

Le paramètre *charset&Encodage* permet d'indiquer le jeu de caractères utilisé dans le corps du message à envoyer, les en-têtes et les noms des fichiers joints. Il définit également le type d'encodage à effectuer sur le corps du message, conformément aux valeurs fournies dans le tableau ci-dessous. Si, par exemple, vous passez 2 ("US-ASCII & 7 bits"), vous indiquez que le jeu de caractères utilisé est de l'ASCII US (utilisation des 128 premiers caractères de la table ASCII exclusivement), et 4D Internet Commands encodera le message sur 7 bits. A noter que la commande **SMTP_SetPrefs** n'effectue pas de conversion dans le jeu de caractères spécifié, c'est à l'utilisateur de s'assurer de la conformité du jeu de caractères. Si vous souhaitez convertir le jeu de caractères utilisé dans un message, vous devez appeler explicitement la commande *SMTP_CharSet*.
Par défaut, ce paramètre a pour valeur 1, les commandes SMTP détectent automatiquement les paramètres appropriés en fonction du contenu du corps du message.

Valeur	Jeu de caractères et encodage du corps	Jeu de caractères des en-têtes et noms de fichiers (encodage base64)
-1	Pas de changement	Pas de changement
0	Application & binary; pas d'encodage	ISO-8859-1
1	Défaut : UTF-8 & base64	Défaut : UTF-8 pour objet, ISO-8859-1 pour les autres champs
2	US-ASCII & 7bit	ISO-8859-1
3	US-ASCII & quotable-printable	ISO-8859-1
4	US-ASCII & base64	ISO-8859-1
5	ISO-8859-1 & quotable-printable	ISO-8859-1
6	ISO-8859-1 & base64	ISO-8859-1
7	ISO-8859-1 & 8bit	ISO-8859-1
8	ISO-8859-1 & binary	ISO-8859-1
9	Réservé	Réservé
10	ISO-2022-JP (Japonais) & 7 bit	ISO-2022-JP
11	ISO-2022-KR (Coréen) & 7 bit	ISO-2022-KR
12	ISO-2022-CN (Chinois traditionnel et simplifié) & 7 bit	ISO-2022-CN
13	HZ-GB-2312 (Chinois simplifié) & 7 bit	HZ-GB-2312
14	Shift-JIS (Japonais) & base64	Shift-JIS
15	UTF-8 & quoted-printable	UTF-8
16	UTF-8 & base64	UTF-8

Note : Nous recommandons d'utiliser les paramétrages par défaut, qui sont adaptés à la plupart des systèmes/applications.

Attention: Le symbole "euro" (€) n'est pas disponible en ISO-8859-1.

Le paramètre *longueurLigne* spécifie la longueur maximale de ligne dans le corps du message. Les commandes SMTP "forcent" le passage à la ligne dans le corps du texte en insérant un retour chariot/retour ligne avant la longueur de ligne maximale lorsque le texte est encodé. Tout nombre peut être spécifié, mais la RFC 2822 indique que la longueur de ligne ne doit pas dépasser 998 et recommande un maximum de 78. Passez -1 pour laisser inchangée la valeur courante.

Par défaut, *longueurLigne* prend la valeur 0 (zéro). Dans ce cas, les commandes SMTP utilisent les valeurs recommandées par les RFC, en fonction du *jeu&encodage*. Les passages à la ligne sont alors insérés selon le tableau suivant :

Type de corps	Passage à la ligne à
Base64	76
Quoted-printable	76
Autres…	pas de passage à la ligne

Le traitement du passage à la ligne est fortement recommandé car de nombreux systèmes et programmes de messagerie ne peuvent gérer les messages contenant des lignes de longueur illimitée. En outre, un courrier passe par de nombreux systèmes avant d'atteindre sa destination finale et un ordinateur situé sur le trajet peut rejeter un message s'il est incapable d'en interpréter le format.

Exemple

Le code suivant envoie un message en UTF-8 encodé en quotable-printable (les en-têtes restent dans leur jeu de caractères par défaut) :

```
$err:=SMTP_SetPrefs(-1;15;-1)
$err:=SMTP_Charset(0;1)  //appliquer les préférences
$err:=SMTP_QuickSend("monmail.com";"monadresse";"destination";"L'euro €";"Le symbole de l'Euro est €")
```

SMTP_Subject

SMTP_Subject (smtp_ID ; msgObjet {; supprimerOption}) -> Résultat

Paramètre	Type		Description
smtp_ID	Entier long	➡	Référence de message
msgObjet	Texte	➡	Objet du message (UTF-8 par défaut)
supprimerOption	Entier	➡	0 = Remplacer sauf si msgObjet vide, 1 = Remplacer, 2 = Supprimer
Résultat	Entier	↩	Code d'erreur

Description

La commande **SMTP_Subject** ajoute l'en-tête "Objet" (Subject) au message référencé par *smtp_ID*. Si un objet avait déjà été défini par une précédente commande *SMTP_Subject*, le nouvel objet écrase le précédent.

smtp_ID contient l'identifiant d'un message électronique créé avec la commande *SMTP_New*.

msgObjet contient un texte concis décrivant le sujet traité en détail dans le corps du message.

Notes :

- Par défaut, l'en-tête "Objet" du message est encodé en UTF-8, ce qui garantit que les caractères envoyés seront correctement interprétés par la quasi totalité des clients de messagerie. Si vous souhaitez utiliser un jeu de caractères spécifique, reportez-vous aux commandes **SMTP_SetPrefs** et **SMTP_Charset**.
- Le texte ne doit pas contenir de retours à la ligne (code ASCII=10). Un retour à la ligne désigne la fin de la section d'en-tête et le début du corps du texte. Les en-têtes suivants risquent alors d'être considérés comme le corps du texte et de ne pas être correctement reconnus par le logiciel du serveur ou du client. Pour plus d'informations sur les zones d'en-tête, veuillez consulter la RFC 822.

Le paramètre optionnel *supprimerOption* vous permet de préciser s'il faut conserver ou supprimer l'en-tête "Objet" éventuellement existant :

- Si vous passez 0 (zéro), le contenu du paramètre *msgObjet* remplace dans la zone d'en-tête tout texte éventuellement présent, sauf si vous passez une chaîne vide dans *msgObjet*, auquel cas l'en-tête "Objet" existant et son contenu sont conservés.
- Si vous passez 1, le contenu du paramètre *msgObjet* remplace dans la zone d'en-tête tout texte éventuellement présent. Dans ce cas, si vous passez une chaîne vide dans *msgObjet*, l'en-tête "Objet" existant est supprimé.
- Si vous passez 2, l'en-tête "Objet" est supprimé du message.
  Si *supprimerOption* est omis, par défaut la valeur 0 est utilisée.

Exemple

Reportez-vous à l'exemple de la commande *SMTP_Body*.

SMTP_To

SMTP_To (smtp_ID ; msgA {; supprimerOption}) -> Résultat

Paramètre	Type		Description
smtp_ID	Entier long	➡	Référence de message
msgA	Texte	➡	Adresse électronique ou liste d'adresses
supprimerOption	Entier	➡	0 = Ajouter, 1 = Remplacer, 2 = Supprimer
Résultat	Entier	↩	Code d'erreur

Description

La commande *SMTP_To* inscrit l'adresse des destinataires principaux d'un message dans la zone d'en-tête "A" (To). Toutes les adresses listées dans les zones d'en-tête "A" et "Cc" d'un message électronique seront visibles par chaque destinataire.

smtp_ID contient l'identifiant d'un message électronique créé avec la commande *SMTP_New*.

msgA contient une ou plusieurs adresses électroniques.

Le paramètre *supprimerOption* vous permet de préciser s'il faut conserver ou supprimer la zone d'en-tête "A" éventuellement existante :

- Si vous passez 0 (zéro), le contenu du paramètre passé est ajouté au contenu de l'en-tête existant.
- Si vous passez 1, le contenu du paramètre passé remplace le contenu de l'en-tête existant. Dans ce cas, si vous avez passé une chaîne vide dans *msgA*, l'en-tête "A" est supprimé.
- Si vous passez 2, l'en-tête "A" est supprimé du message.
- Si *supprimerOption* est omis, par défaut la valeur 0 est utilisée.

Exemple

Voir l'exemple de la commande *SMTP_Body*.

IC TCP/IP

- Routines de bas niveau, Présentation
- TCP_Close
- TCP_Listen
- TCP_Open
- TCP_Receive
- TCP_ReceiveBLOB
- TCP_Send
- TCP_SendBLOB
- TCP_State

Routines de bas niveau, Présentation

TCP/IP ou Transmission Control Protocol/Internet Protocol est le principal protocole utilisé pour l'envoi de données sur Internet. Les commandes Internet de bas niveau de 4D permettent d'établir des sessions TCP puis d'envoyer et/ou de recevoir des paquets TCP via ces sessions.

Il existe deux manières d'établir une connexion TCP :

- La première consiste à exécuter la commande *TCP_Open*. Celle-ci ouvre une connexion avec le domaine et sur le port spécifiés. Elle permet de se connecter à un serveur TCP. La commande *TCP_Open* permet l'utilisation du protocole SSL (Secured Socket Layer) afin de sécuriser la connexion.
- La seconde consiste à exécuter la commande *TCP_Listen*. Celle-ci ouvre une connexion avec le domaine et sur le port spécifiés, et écoute les connexions entrantes. La meilleure façon de déterminer si une connexion a été établie consiste à vérifier l'état de la session avec la commande *TCP_State*, après l'exécution de *TCP_Listen*. Un code de statut est alors renvoyé, indiquant l'état courant de la session. A partir de là, vous pouvez envoyer et/ou recevoir des paquets TCP comme vous le feriez lors d'une connexion établie avec *TCP_Open*.

Dans tous les cas, toute connexion TCP ouverte doit être refermée à l'aide de la commande *TCP_Close*.

Les commandes TCP/IP de bas niveau nécessitent une connaissance approfondie des protocoles de communication. Pour obtenir des informations complémentaires sur les différents numéros de port affectés à TCP/IP, les protocoles de communication, les impératifs d'adressage, etc., veuillez vous reporter aux RFCs.

Références de connexions dans les commandes TCP

Les commandes Internet de 4D permettent de passer directement une référence de connexion POP3, IMAP ou FTP aux commandes TCP de bas niveau et inversement.
En effet, d'une part les protocoles évoluent régulièrement, donnant naissance à de nouvelles commandes ; d'autre part, certains progiciels font leur propre interprétation des RFCs — rendant les implémentations normalisées inutilisables. Avec ce principe, le développeur pourra créer lui-même les fonctions de haut niveau dont il a besoin (à la place de fonctions existantes ou pour palier à l'inexistence d'une fonction) à l'aide des commandes bas niveau TCP.
Cette fonctionnalité accroît significativement l'ouverture et les possibilités de développement puisque les développeurs peuvent créer leurs propres commandes de haut niveau sans devoir réécrire toutes les commandes nécessaires à l'exploitation d'un protocole.

Dans cet exemple, la commande *IMAP_Capability* est remplacée par une fonction équivalente développée à l'aide des commandes TCP_IP.

- Voici la méthode initiale utilisant la commande *IMAP_Capability* :

```
$ErrorNum:=IMAP_Login(vHost;vUserName;vUserPassword;vImap_ID)
If($ErrorNum=0)
   C_TEXT(vCapability)
   $ErrorNum:=IMAP_Capability(vImap_ID;vCapability)
   ... ` Commandes IMAP utilisant le paramètre vImap_ID
End if
$ErrorNum:=IMAP_Logout(vImap_ID)
```

- Cette méthode peut être remplacée par :

```
$ErrorNum:=IMAP_Login(vHost;vUserName;vUserPassword;vImap_ID)
If($ErrorNum =0)
   C_TEXT(vCapability)
  ` Méthode TCP utilisant la valeur du paramètre vImap_ID :
   $ErrorNum:=My_IMAP_Capability(vImap_ID)
   ... ` Commandes IMAP utilisant le paramètre vImap_ID
```

```
End if
$ErrorNum:=IMAP_Logout(vImap_ID)
```

- Voici le code de la fonction ***My_IMAP_Capability*** :

```
C_LONGINT($1;$vErrorNum;$0)
C_TEXT($vTexteEnvoyé;$vTexteReçu;vCapability)
C_TEXT($2)

$vImap_Id:=$1
$vCmd_Id:="A001" ` Cet ID de commande doit être unique (cf. RFC 2060)
$MyvtRequestCmd:="CAPABILITY"
$vTexteEnvoyé:=$vCmd_Id+""+$MyvtRequestCmd+Char(13)+Char(10)
$vTexteReçu:=""
$vErrorNum:=TCP_Send($vImap_Id;$vTexteEnvoyé)
If($vErrorNum=0)
   $vErrorNum:=TCP_Receive($vImap_Id;$vTexteReçu)
   Case of
      :($vErrorNum#0) `Erreur de réception
         vCapability:=""
      :(Position($vCmd_Id+" OK ";$vTexteReçu)#0)
 ` Exécution de la commande avec succès
         vCapability:=$vTexteReçu
 ` Dans cet exemple, nous ne traitons pas la chaîne reçue
      :(Position($vCmd_Id+" BAD ";$vTexteReçu)#0)
 ` Echec de l'exécution de la commande (erreur de syntaxe
 ` ou commande inconnue)
         vCapability:=""
         $vErrorNum:=10096
   End case
End if
$0:=$vErrorNum
```

TCP_Close

TCP_Close (tcp_ID) -> Résultat

Paramètre	Type		Description
tcp_ID	Entier long	➡	Référence d'une session TCP ouverte
		⬅	0 = la session a été correctement refermée
Résultat	Entier	↩	Code d'erreur

Description

La commande *TCP_Close* referme la session TCP référencée par *tcp_ID*.

Lorsqu'une session TCP est ouverte, elle occupe l'un des 64 "canaux" (streams) disponibles. Si les sessions ne sont jamais refermées, lorsque leur nombre atteint 64 l'utilisateur ne peut plus en ouvrir d'autre.

tcp_ID contient la référence d'une session TCP ouverte avec la commande *TCP_Open* ou *TCP_Listen*. Après exécution de la commande, si la session a été correcterment refermée, ce paramètre prend la valeur 0 (zéro).

TCP_Listen

TCP_Listen (adresseIP ; portDistant ; portLocal ; timeOut ; tcp_ID) -> Résultat

Paramètre	Type		Description
adresseIP	Chaîne	➡	Adresse IP locale d'écoute ou "" pour écouter sur toutes les adresses disponibles
		⬅	Adresse IP distante utilisée (si une variable contenant une chaîne vide est passée)
portDistant	Entier	➡	*** Paramètre ignoré ***
portLocal	Entier	➡	Numéro de port local, 0 = Utiliser un port local vacant
		⬅	Numéro du port local utilisé (si 0 passé)
timeOut	Entier	➡	Nombre de secondes à attendre avant timeout, 0 = Pas de timeout
tcp_ID	Entier long	⬅	Référence de la session TCP ouverte
Résultat	Entier	↩	Code d'erreur

Description

La commande *TCP_Listen* ouvre une "socket" de communication sur le port défini par les paramètres *adresseIP* et *portLocal*.

Cette commande ne rend pas la main à la méthode d'appel de 4D tant que la connexion n'a pas été ouverte ou que la période de *timeOut* ne s'est pas écoulée. L'exécution de *TCP_Listen* peut donc provoquer temporairement le blocage de la base. Toutefois, cette commande est particulièrement adaptée au multi-process : elle partage le temps CPU avec les autres process 4D en cours d'exécution.
Il est donc conseillé d'exécuter *TCP_Listen* au sein d'un process 4D particulier (en particulier si aucun *timeOut* n'est spécifié).

Le paramètre *adresseIP* contient l'adresse IP utilisée pour la connexion:

- Vous pouvez passer l'adresse locale sur laquelle doit être effectuée la connexion entrante.
- Si vous passez une chaîne vide, la commande écoute sur toutes les adresses disponibles de la machine.
- Si vous passez une variable contenant une chaîne vide, la commande retournera en outre dans le paramètre l'adresse IP distante utilisée pour la connexion.

Le paramètre *portLocal* contient le numéro de port TCP à utiliser pour la communication. Si vous passez 0 (zéro), la commande utilisera un port vacant et retournera son numéro dans ce paramètre.

Le paramètre *timeOut* spécifie le nombre maximum de secondes pendant lequel la commande attendra une connexion entrante. Si vous passez 0 (zéro), la commande attendra indéfiniment la connexion (pas de timeout). Utilisez cette option avec précaution car le contrôle n'est pas restitué au process d'appel 4D si aucune connexion n'est effectuée. En particulier, ne passez pas 0 dans ce paramètre si la base est mono-process.

Le paramètre *tcp_ID* retourne la référence de la session ouverte. Cette référence sera utilisée par toutes les commandes TCP ultérieures exécutées dans la session.

Toute connexion TCP ouverte par la commande *TCP_Listen* doit être finalement refermée à l'aide de la commande *TCP_Close*.

Exemple

```
C_LONGINT(vTCPID)
C_LONGINT(vStatus)
$err:=TCP_Listen("";0;0;30;vTCPID)
$err:=TCP_State(vTCPID;vStatut)
If(vStatut=2)  //La socket est ouverte et à l'écoute
   FaireQuelqueChose
   $err:=TCP_Close(vTCPID)
End if
```

TCP_Open

TCP_Open (nomServeur ; portDistant ; tcp_ID ; paramsSession) -> Résultat

Paramètre	Type		Description
nomServeur	Chaîne	→	Nom ou adresse IP du serveur
portDistant	Entier	→	Port distant auquel se connecter (0 = indifférent)
tcp_ID	Entier long	←	Référence de la session TCP ouverte
paramsSession	Entier	→	Paramètres de la session TCP 0 = Synchrone (Valeur par défaut), 1 = Asynchrone, 2 = Mode SSL synchrone, 3 = Mode SSL asynchrone
Résultat	Entier	↩	Code d'erreur

Description

La commande *TCP_Open* ouvre une connexion TCP sortante vers un domaine.

TCP_Open établit une connexion avec *nomServeur*, sur le port référencé par *portDistant* (si ce paramètre est différent de 0), et retourne le numéro de la session dans *tcp_ID*. Cette valeur sera utilisée par tous les appels TCP ultérieurs se rapportant à cette session. Par défaut, une session ouverte par *TCP_Open* n'est maintenue que pendant 30 secondes (timeout) si aucune donnée n'est reçue. Cette valeur de timeout par défaut peut être modifiée à l'aide de la commande *IT_SetTimeOut*.

Le paramètre *nomServeur* contient le nom ou l'adresse IP du serveur avec lequel vous ouvrez une connexion.

Le paramètre *portDistant* permet d'indiquer le port TCP de la machine désignée par *nomServeur*, avec laquelle vous souhaitez établir une connexion. Pour utiliser n'importe quel port, passez 0 (zéro) dans ce paramètre.

Note : Après un appel à *TCP_Open* (ou *TCP_Listen*), *portDistant* peut retourner une valeur négative si la valeur passée dans ce paramètre est supérieure à 32767. Cela ne perturbe pas la connexion. Toutefois, pour éviter cette situation, vous pouvez utiliser une variable intermédiaire :

```
$v_ portDistant:=v_portDistant
$err:=TCP_Open(v_AdrIPServeurDistant;$v_portDistant;v_SessionID)
```

tcp_ID retourne la référence de la session ouverte. Cette référence sera utilisée par toutes les commandes TCP ultérieures exécutées dans la session.

Le paramètre optionnel *paramsSession* vous permet de définir si vous souhaitez que la session TCP soit ou non sécurisée via SSL, et si vous souhaitez que les commandes TCP partagent ou non le temps CPU avec les autres process de 4D (mode asynchrone ou synchrone). Ce paramètre est pris en compte par toutes les commandes TCP exécutées dans la session courante.
paramsSession accepte les valeurs suivantes :

- 0 = Mode standard synchrone (mode par défaut, fonctionnement identique à celui des versions précédentes de 4D Internet Commands).
- 1 = Mode standard asynchrone
- 2 = Mode SSL synchrone. Toutes les commandes TCP référençant cette session (*tcp_ID*) seront exécutées en mode synchrone et utiliseront le protocole SSL.
- 3 = Mode SSL asynchrone. Toutes les commandes TCP référençant cette session (*tcp_ID*) seront exécutées en mode asynchrone et utiliseront le protocole SSL.

Le protocole SSL (Secured Socket Layer) permet d'établir des communications TCP sécurisées (reportez-vous à la documentation de 4D pour plus d'informations sur l'installation et la mise en oeuvre du protocole SSL).

Note : L'erreur 10089 est retournée lorsque vous passez les valeurs 2 ou 3 si la connexion SSL ne peut pas être ouverte (librairie SLI manquante dans le dossier 4D Extensions).

Toute connexion TCP ouverte par la commande *TCP_Open* doit être finalement refermée à l'aide de la commande *TCP_Close*.

Asynchrone/Synchrone

- En mode **asynchrone**, les commandes Internet de 4D rendent la main au moteur de 4D immédiatement après leur exécution, sans attendre la fin du process de connexion (c'est-à-dire, sans

attendre que la connexion avec le serveur distant soit établie). Le mode asynchrone est utile lorsque vous souhaitez que les commandes TCP ne consomment pas le temps machine de 4D.
- En mode **synchrone**, les commandes Internet de 4D ne rendent la main au moteur de 4D (c'est-à-dire aux autres process de 4D) que lorsque le process de connexion a pris fin (que la connexion ait réussi ou non).

Exemple

Vous souhaitez vous connecter à un site Web en Https ; assurez-vous que la librairie SLI est correctement installée et ouvrez la connexion sur le port 443 :

```
$vError:=TCP_Open(hostName;443;tcp_ID;2)
...
$vError:=TCP_Close(tcp_ID)  `Ne pas oublier de refermer la session
```

TCP_Receive

TCP_Receive (tcp_ID ; texte) -> Résultat

Paramètre	Type		Description
tcp_ID	Entier long	➡	Référence à une session TCP ouverte
texte	Texte	⬅	Texte reçu
Résultat	Entier	⮌	Code d'erreur

Description

La commande *TCP_Receive* permet de recevoir des paquets de données au cours d'une session TCP.

Passez dans *tcp_ID* la référence d'une session TCP ouverte avec la commande *TCP_Open* ou *TCP_Listen*.

Le paramètre *texte* retourne sous forme de texte les données reçues.

Lors de la réception de données par paquets TCP, vous ne pouvez pas avoir la certitude que la totalité des données sont reçues par un seul appel *TCP_Receive*. La commande *TCP_Receive* est donc généralement appelée dans une boucle '**Repeter**' qui vérifie en permanence le statut de la connexion ou attend une valeur particulière.

Exemple

```
C_LONGINT($tcp_id)
C_TEXT($pageweb;$tampon)
C_LONGINT(vEtat;$erreur)
$pageweb:=""
vEtat:=0
Repeat
   $erreur:=TCP_Receive($tcp_id;$tampon)
   $erreur:=TCP_State($tcp_id;vEtat)
   $pageweb:=$pageweb+$tampon
Until((vEtat=0)|($erreur#0))  //Jusqu'à ce que le serveur ferme la connexion,
 //ou qu'une erreur soit retournée
```

TCP_ReceiveBLOB

TCP_ReceiveBLOB (tcp_ID ; donnéesReçues) -> Résultat

Paramètre	Type		Description
tcp_ID	Entier long	➡	Référence à une session TCP ouverte
donnéesReçues	BLOB	⬅	BLOB devant recevoir les données
Résultat	Entier	↩	Code d'erreur

Description

La commande *TCP_ReceiveBLOB* permet de recevoir des paquets de données au cours d'une session TCP. Son fonctionnement est semblable à celui de la commande *TCP_Receive*, à la différence près qu'elle reçoit des données de type BLOB au lieu d'un texte. Ce principe permet de s'affranchir de la limite des 32 000 caractères inhérente aux données de type Texte, et de recevoir des données binaires.

Passez dans *tcp_ID* la référence d'une session TCP ouverte avec la commande *TCP_Open* ou *TCP_Listen*.

Le paramètre *donnéesReçues* retourne les données reçues dans un BLOB.
Lors de la réception de données par paquets TCP, vous ne pouvez pas avoir la certitude que la totalité des données ont été reçues via un seul appel à *TCP_ReceiveBLOB*. La commande *TCP_ReceiveBLOB* est donc généralement utilisée dans une boucle du type **Repeter...Jusque** qui vérifie en permanence le statut de la connexion ou attend une valeur particulière.

Exemple

Voici la structure type d'une méthode utilisant la commande *TCP_ReceiveBLOB* :

```
C_BLOB($Blob_Reçu;$Blob_Concaténé)
C_LONGINT($srcpos;$dstpos)
Repeat
   $Err:=TCP_ReceiveBLOB($TCP_ID;$Blob_Reçu)
   $Err:=TCP_State($TCP_ID;$State)
   $srcpos:=0
   $dstpos:=BLOB size($Blob_Concaténé)
  `Concaténation des données reçues
   COPY BLOB($Blob_Reçu;$Blob_Concaténé;$srcpos;$dstpos;BLOB size($Blob_Reçu))
Until(($State=0)|($Err#0))
```

TCP_Send

TCP_Send (tcp_ID ; texteAEnvoyer) -> Résultat

Paramètre	Type		Description
tcp_ID	Entier long	➡	Référence de session TCP ouverte
texteAEnvoyer	Texte	➡	Texte à envoyer
Résultat	Entier	⮌	Code d'erreur

Description

La commande **TCP_Send** envoie des données à la session TCP désignée par *tcp_ID*.

tcp_ID est la référence d'une session TCP ouverte, établie avec la commande **TCP_Open** ou **TCP_Listen**.

Le paramètre *texteAEnvoyer* contient une valeur de type texte à envoyer à la session TCP référencée par *tcp_ID*.

Note : En cas d'envoi de texte non us-ascii ou de texte de grande taille, il est recommandé d'utiliser la commande **TCP_SendBLOB** car elle permet un meilleur contrôle de l'encodage du texte.

TCP_SendBLOB

TCP_SendBLOB (tcp_ID ; blobAEnvoyer) -> Résultat

Paramètre	Type		Description
tcp_ID	Entier long	→	Référence à une session TCP ouverte
blobAEnvoyer	BLOB	→	Données à envoyer
Résultat	Entier	↩	Code d'erreur

Description

La commande *TCP_SendBLOB* envoie des données à la session TCP désignée par *tcp_ID*. Elle fonctionne de manière semblable à la commande *TCP_Send*, à la différence près qu'elle envoie des données de type BLOB au lieu d'un texte. Ce fonctionnement permet de s'affranchir de la limite des 32 000 caractères inhérente aux données de type Texte, et ainsi d'envoyer des données binaires.

tcp_ID est la référence d'une session TCP ouverte, établie avec la commande *TCP_Open* ou *TCP_Listen*.

Le paramètre *blobAEnvoyer* contient les données de type BLOB à envoyer à la session TCP référencée par *tcp_ID*.

Note sur l'indépendance de plate-forme : Lors de l'envoi de données binaires dans un format propriétaire, il vous appartient de traiter les conversions d'octets ("byte swapping") entre les différentes plates-formes, si nécessaire.

Exemple

Cet exemple place du texte dans un BLOB puis l'envoie dans la session TCP :

```
C_BLOB($Blob_AEnvoyer)
C_TEXT(v_Txt_Send)
TEXT TO BLOB(v_Txt_Send;$Blob_AEnvoyer;Mac text without length;*)
$err:=TCP_SendBLOB(v_tcp_ID;$Blob_AEnvoyer)
```

TCP_State

TCP_State (tcp_ID ; codeStatut) -> Résultat

Paramètre	Type		Description
tcp_ID	Entier long	➡	Référence d'une session TCP ouverte
codeStatut	Entier	⬅	Code du statut TCP
Résultat	Entier	↩	Code d'erreur

Description

La commande *TCP_State* renvoie une valeur indiquant le statut d'une connexion TCP particulière.
tcp_ID contient la référence d'une session TCP ouverte avec la commande *TCP_Open* ou *TCP_Listen*.
Le paramètre *codeStatut* retourne l'un des codes de statut suivants :

0 Connexion fermée
2 A l'écoute d'une connexion entrante
8 Connexion établie

Exemple

Cet exemple suppose qu'une connexion TCP valide a été établie et est identifiée par la variable $tcp_id. Dans cet exemple, une commande est envoyée à un serveur Web pour demander une page d'informations et les résultats sont récupérés dans une boucle de type **Repeter**. Comme les serveurs Web referment automatiquement les connexions dès qu'ils ont terminé leur tâche, la méthode continuera à recevoir des données jusqu'à ce que la connexion soit stoppée ou qu'une erreur se produise.

```
C_LONGINT($tcp_id)
C_LONGINT(vEtat;$err)
C_TEXT($commande;$tampon;$réponse)
If(TCP_Send($tcp_id;$commande)=0)
   vEtat:=0
   Repeat
      $err:=TCP_Receive($tcp_id;$tampon)
      $err:=TCP_State($tcp_id;vEtat)
      $réponse:=$réponse+$tampon
   Until((vEtat=0)|($err#0))
End if
```

IC UDP

- Commandes UDP, Présentation
- UDP_Delete
- UDP_New
- UDP_ReceiveBLOBFrom
- UDP_ReceiveFrom
- UDP_SendBLOBTo
- UDP_SendTo

Commandes UDP, Présentation

UDP (User Datagram Protocol) est un protocole de communication réseau déconnecté permettant la transmission de données par paquets. Facile à mettre en oeuvre, il est plus rapide et plus simple que le protocole TCP (l'en-tête UDP tient sur 8 octets, à comparer aux 20 octets minimum de l'en-tête TCP), mais il n'offre pas le même niveau de fiabilité. Il est adapté aux applications privilégiant la rapidité de transmission des données. En revanche, il ne permet pas de vérifier que les paquets sont bien arrivés à destination et ne propose pas de contrôle d'erreur ni de système de récupération.

Exemple

L'exemple suivant utilise le protocole UDP pour récupérer dans des tableaux la liste de tous les postes 4D Server actifs sur le réseau local:

```
ARRAY TEXT(taHost;0)
ARRAY TEXT(taNomMachine;0)
ARRAY TEXT(taService;0)
ARRAY TEXT(taNomDB;0)
C_BLOB($Blob)

$Addr:="255.255.255.255"
$Port:=19813
$Offset:=32
SET BLOB SIZE($Blob;96;0)
TEXT TO BLOB("4D Server II";$Blob;Mac text without length;$Offset)

$Err:=UDP_New(0;$udpRef)
$Err:=UDP_SendBLOBTo($udpRef;$Addr;$Port;$Blob)
$Secs:=5
$Timeout:=Milliseconds+($Secs*1000)
Repeat
   DELAY PROCESS(Current process;6)  `... en ticks
   SET BLOB SIZE($Blob;0;0)
   $PeerAddr:=$Addr
   $Err:=UDP_ReceiveBLOBFrom($udpRef;$PeerAddr;$Port;$Blob)

   If(BLOB size($Blob)>0)
      $Offset:=0
      $Host:=BLOB to text($Blob;Mac C string;$Offset;32)
      $Offset:=32
      $Service:=BLOB to text($Blob;Mac C string;$Offset;32)
      $Offset:=64
      $NomDB:=BLOB to text($Blob;Mac C string;$Offset;32)
      $Pos:=Find in array(taNomMachine;$Host)
      If($Pos=-1)
         APPEND TO ARRAY(taHost;$PeerAddr)
         APPEND TO ARRAY(taNomMachine;$Host)
         APPEND TO ARRAY(taService;$Service)
         APPEND TO ARRAY(taNomDB;$NomDB)
      End if
   End if
Until((Milliseconds>$Timeout)|($Err#0))
$Err:=UDP_Delete($udpRef)
```

UDP_Delete

UDP_Delete (udp_ID) -> Résultat

Paramètre	Type		Description
udp_ID	Entier long	⇒	Référence de connexion UDP
Résultat	Entier	⇐	Code d'erreur

Description

La commande *UDP_Delete* referme la connexion UDP référencée par *udp_ID*, préalablement créée à l'aide de la commande *UDP_New*.

UDP_New

UDP_New (portLocal ; udp_ID) -> Résultat

Paramètre	Type		Description
portLocal	Entier	⇒	Numéro de port local, 0 = Utiliser un port local vacant
udp_ID	Entier long	⇐	Référence de connexion UDP
Résultat	Entier	⇒	Code d'erreur

Description

La commande *UDP_New* permet de créer une connexion (socket) UDP sur le port passé dans le paramètre *portLocal*. Pour utiliser n'importe quel port local vacant, passez 0 dans *portLocal*.

La référence de la connexion UDP ouverte est retournée dans le paramètre *udp_ID*.

Lorsque cette connexion n'est plus nécessaire, n'oubliez pas de la supprimer à l'aide de la commande *UDP_Delete* afin de libérer la mémoire.

UDP_ReceiveBLOBFrom

UDP_ReceiveBLOBFrom (udp_ID ; nomServeur ; portDistant ; blob) -> Résultat

Paramètre	Type		Description
udp_ID	Entier long	➡	Référence de connexion UDP
nomServeur	Variable texte	⬅	Nom ou adresse IP du serveur qui répond
portDistant	Variable entier long	⬅	Port du serveur distant qui répond
blob	BLOB	⬅	BLOB reçu
Résultat	Entier	↩	Code d'erreur

Description

La commande **UDP_ReceiveBLOBFrom** permet de recevoir un BLOB envoyé via la connexion UDP désignée par le paramètre *udp_ID*.

Passez des variables dans les paramètres *nomServeur* et *portDistant*. A l'issue de l'exécution de la commande, ces variables contiennent respectivement le nom (ou l'adresse IP) et le numéro du port du serveur distant duquel le BLOB a été récupéré.

Le BLOB récupéré est retourné dans la variable *blob*.

UDP_ReceiveFrom

UDP_ReceiveFrom (udp_ID ; nomServeur ; portDistant ; texte) -> Résultat

Paramètre	Type		Description
udp_ID	Entier long	→	Référence de connexion UDP
nomServeur	Variable texte	←	Nom ou adresse IP du serveur qui répond
portDistant	Variable entier long	←	Port du serveur distant qui répond
texte	Texte	←	Texte reçu
Résultat	Entier	↩	Code d'erreur

Description

La commande **UDP_ReceiveFrom** permet de recevoir du texte envoyé via la connexion UDP désignée par le paramètre *udp_ID*.

Passez des variables dans les paramètres *nomServeur* et *portDistant*. A l'issue de l'exécution de la commande, ces variables contiennent respectivement le nom (ou l'adresse IP) et le numéro du port du serveur distant duquel le texte a été récupéré.

Le texte récupéré est retourné dans la variable *texte*.

UDP_SendBLOBTo

UDP_SendBLOBTo (udp_ID ; nomServeur ; portDistant ; blobAEnvoyer) -> Résultat

Paramètre	Type		Description
udp_ID	Entier long	➡	Référence de connexion UDP
nomServeur	Chaîne	➡	Nom ou adresse IP du serveur
portDistant	Entier	➡	Port distant à utiliser (0 = indifférent)
blobAEnvoyer	BLOB	➡	BLOB à envoyer
Résultat	Entier	⮌	Code d'erreur

Description

La commande *UDP_SendBLOBTo* permet d'envoyer un BLOB via la connexion UDP ouverte référencée dans le paramètre *udp_ID*.

Le paramètre *nomServeur* contient le nom ou l'adresse IP du serveur auquel les données doivent être envoyées.

Le paramètre *portDistant* permet d'indiquer le port de la machine distante à utiliser. Pour utiliser n'importe quel port, passez 0 (zéro) dans ce paramètre.

Le paramètre *blobAEnvoyer* contient le BLOB à transmettre.

UDP_SendTo

UDP_SendTo (udp_ID ; nomServeur ; portDistant ; texteAEnvoyer) -> Résultat

Paramètre	Type		Description
udp_ID	Entier long	⇒	Référence de connexion UDP
nomServeur	Chaîne	⇒	Nom ou adresse IP du serveur
portDistant	Entier	⇒	Port distant à utiliser (0 = indifférent)
texteAEnvoyer	Texte	⇒	Texte à envoyer
Résultat	Entier	⇐	Code d'erreur

Description

La commande *UDP_SendTo* permet d'envoyer des données texte via la connexion UDP ouverte référencée dans le paramètre *udp_ID*.

Le paramètre *nomServeur* contient le nom ou l'adresse IP du serveur auquel les données doivent être envoyées.

Le paramètre *portDistant* permet d'indiquer le port de la machine distante à utiliser. Pour utiliser n'importe quel port, passez 0 (zéro) dans ce paramètre.

Le paramètre *texteAEnvoyer* contient le texte à transmettre.

IC Utilities

- Utilitaires, Présentation
- IT_Decode
- IT_Encode
- IT_ErrorText
- IT_GetPort
- IT_GetProxy
- IT_GetTimeOut
- IT_MyTCPAddr
- IT_Platform
- IT_PPPConnect
- IT_PPPDisconnect
- IT_PPPStatus
- IT_SetPort
- IT_SetProxy
- IT_SetTimeOut
- IT_TCPversion
- IT_Version
- *IT_MacTCPInit*
- *IT_MacTCPVer*

Utilitaires, Présentation

Cette section regroupe un ensemble de routines utilitaires. Certaines de ces commandes permettent de déterminer l'environnement dans lequel fonctionnent les machines des utilisateurs, les versions des logiciels utilisés ou l'état courant et l'adresse IP de leur ordinateur.
Les autres commandes permettent de traiter les codes d'erreur, d'encoder et de décoder des fichiers, et d'agir sur la valeur d'attente maximale (timeout) par défaut applicable à la plupart des commandes des autres thèmes.

IT_Decode

IT_Decode (nomFichier ; nomFichierDécodé ; modeDécodage) -> Résultat

Paramètre	Type		Description
nomFichier	Texte	➡	Chemin d'accès local à un fichier encodé
nomFichierDécodé	Texte	➡	Chemin d'accès du fichier décodé
		⬅	Chemin d'accès résultant du fichier décodé
modeDécodage	Entier	➡	Mac & Win: 0 = Pas d'encodage (DataFork uniquement) ±1 = BinHex ±2 = Base64 (n'envoie que la DataFork) ±7 = UUEncode; Mac seulement: ±3 = AppleSingle ±4 = AppleDouble ±5 = AppleSingle ET Base64 ±6 = AppleDouble ET Base64
Résultat	Entier	↩	Code d'erreur

Description

La commande *IT_Decode* décode un fichier au moyen du *modeDécodage* spécifié. Le fichier original n'est pas modifié, une copie décodée est créée.

Le paramètre *nomFichier* contient le chemin d'accès complet du fichier que vous voulez décoder.

Le paramètre *nomFichierDécodé* peut contenir :

- Le chemin d'accès complet de la copie decodée du fichier, ce qui vous permet de définir son nom et son emplacement.
- Le chemin d'accès complet du dossier devant recevoir la copie décodée du fichier, sans spécification du nom de fichier, auquel cas le nom du fichier décodé créé est le nom initial du fichier.
- Une chaîne vide, auquel cas la commande *IT_Decode* attribue un nom par défaut au document et le place dans le même répertoire que le fichier spécifié dans le premier paramètre.
  Qu'il soit spécifié ou non, le chemin d'accès final du document décodé est retourné dans le paramètre *nomFichierDécodé*.

Le paramètre *modeDécodage* vous permet d'indiquer la méthode de décodage à appliquer au fichier. La valeur par défaut est 1 (décodage binhex). Les méthodes de décodage proposées sont les suivantes :

Code	Méthode	Disponible sur
1	BinHex	Win & Mac
2	Base64 (Data fork uniquement)	Win & Mac
3	AppleSingle	Mac
4	AppleDouble	Mac
5	AppleSingle et Base64	Mac
6	AppleDouble et Base64	Mac
7	UUEncode	Win & Mac
8	MacBinary	Win & Mac

Pour le décodage au moyen d'AppleDouble (modes de décodage 4 & 6), la commande recherche la partie ressources dans un fichier nommé "%nomFichier".

IT_Encode

IT_Encode (nomFichier ; nomFichierCodé ; modeCodage) -> Résultat

Paramètre	Type		Description
nomFichier	Texte	⇒	Chemin d'accès local à un fichier
nomFichierCodé	Texte	⇒	Chemin d'accès du fichier encodé
		⇐	Chemin d'accès au fichier encodé résultant
modeCodage	Entier	⇒	Mac & Win: 0 = Pas d'encodage (DataFork uniquement) ±1 = BinHex ±2 = Base64 (n'envoie que la DataFork) ±7 = UUEncode; Mac seulement: ±3 = AppleSingle ±4 = AppleDouble ±5 = AppleSingle ET Base64 ±6 = AppleDouble ET Base64
Résultat	Entier	↩	Code d'erreur

Description

La commande *IT_Encode* encode un fichier au moyen du *modeCodage* spécifié. Le fichier désigné n'est pas modifié, seule une copie encodée est créée. Le nom du fichier encodé créé est le nom initial du fichier plus un suffixe ajouté pour spécifier la méthode de codage. Pour le codage Binhex, le suffixe ".hqx" est ajouté. Pour le codage Base64, le suffixe ".b64" est ajouté. Pour le codage AppleSingle, le suffixe ".as" est ajouté.

Le paramètre *nomFichier* contient le chemin d'accès complet au fichier que vous voulez encoder.

Le paramètre *nomFichierCodé* peut contenir :

- Le chemin d'accès complet de la copie encodée du fichier, ce qui vous permet de définir son nom et son emplacement.
- Le chemin d'accès complet du dossier devant recevoir la copie encodée, sans spécification du nom de fichier, auquel cas le nom du fichier encodé créé est le nom initial du fichier plus un suffixe ajouté pour spécifier la méthode de codage.
- Une chaîne vide, auquel cas *IT_Encode* attribue un nom par défaut au document (défini suivant les principes énoncés précédemment) et le place dans le même dossier que le fichier désigné dans le premier paramètre.
  Qu'il soit spécifié ou non, le chemin d'accès final du document encodé est retourné dans le paramètre *nomFichierCodé*. En raison des conflits de noms potentiels dans le répertoire spécifié, il est conseillé de prendre en compte la valeur résultante du paramètre.

Le paramètre *modeCodage* vous permet de définir la méthode d'encodage à appliquer au fichier. La valeur par défaut est 1 (encodage binhex). Les méthodes d'encodage proposées sont les suivantes :

Code	Méthode	Disponible sur
1	BinHex	Win & Mac
2	Base64 (Data fork uniquement)	Win & Mac
3	AppleSingle	Mac
4	AppleDouble	Mac
5	AppleSingle et Base64	Mac
6	AppleDouble et Base64	Mac
7	UUEncode	Win & Mac
8	MacBinary	Win & Mac

Pour le codage au moyen d'AppleDouble (modes de codage 4 & 6), deux fichiers sont créés, appelés "%nomFichier" et "nomFichier".

IT_ErrorText

IT_ErrorText (numErreur) -> Résultat

Paramètre	Type		Description
numErreur	Entier	→	Code d'erreur
Résultat	Chaîne	↩	Texte de l'erreur

Description

La commande *IT_ErrorText* retourne le libellé de l'erreur dont le numéro a été passé en paramètre. Notez qu'il s'agit d'une des rares commandes Internet de 4D qui ne retourne pas un entier comme valeur.
Le paramètre *erreur* contient le numéro de l'erreur.

Exemple

La méthode ***VérifErreur*** suivante affiche un message d'alerte expliquant l'erreur. Cette méthode reçoit deux paramètres : le nom de la commande (*$Commande*) et la valeur de l'erreur (fournie par l'exécution de la commande dans le paramètre de la méthode). ***VérifErreur*** renvoie une valeur booléenne indiquant si la commande a retourné le code d'erreur 0 (zéro). Si ce n'est pas le cas, la valeur retournée (*$0*) est **Faux**, sinon **Vrai**.

```
  `Méthode VérifErreur ("NOM DE COMMANDE"; Erreur# ) -> Vrai/Faux
C_TEXT(vMsgErreur)
$Commande:=$1
$Erreur:=$2
$Resultat:=True
If($Erreur#0)
   $Resultat:=False
   vMsgErreur:=IT_ErrorText($Erreur)
   ALERT("ERREUR -- "+Char(13)+"Commande: "+$Commande+Char(13)+"Code
d'erreur:"+String($Erreur)+Caractere(13)+"Description: "+vMsgErreur)
End if
$0:=$Resultat
```

IT_GetPort

IT_GetPort (protocole ; port) -> Résultat

Paramètre	Type		Description
protocole	Entier	⇒	1 = FTP ; 2 = SMTP ; 3 = POP3 ; 4 = IMAP ; 12 = SMTP SSL ; 13 = POP3 SSL ; 14 = IMAP SSL
port	Entier	⇐	Numéro de port
Résultat	Entier	⇒	Code d'erreur

Description

Pour le *protocole* spécifié, la commande *IT_GetPort* récupère le numéro du *port* courant utilisé par les commandes Internet de 4D relatives à ce protocole.

IT_GetProxy

IT_GetProxy (protocole ; typeProxy ; nomServeurProxy ; portProxy ; idUtilisateurProxy) -> Résultat

Paramètre	Type		Description
protocole	Entier	→	1 = FTP ; 2 = SMTP ; 3 = POP3 ; 4 = IMAP
typeProxy	Entier	←	0 = Aucun ; 1 = SOCKS
nomServeurProxy	Chaîne	←	Nom ou adresse IP du serveur proxy SOCKS
portProxy	Entier	←	Port du proxy auquel se connecter
idUtilisateurProxy	Texte	←	ID d'utilisateur pour SOCKS
Résultat	Entier	↩	Code d'erreur

Description

La commande *IT_GetProxy* retourne les paramètres courants appliqués au routage, via un serveur proxy SOCKS, du protocole spécifié. Les valeurs sont celles par défaut à moins qu'un appel antérieur à *IT_SetProxy* ne les ait modifiées. Pour plus d'informations sur ces paramètres, reportez-vous à la description de la commande *IT_SetProxy*.

Le paramètre *protocole* vous permet de spécifier le protocole à filtrer. Passez 1 pour le protocole FTP, 2 pour le protocole SMTP, 3 pour le protocole POP3 et 4 pour le protocole IMAP.

Le paramètre *typeProxy* retourne un code indiquant si un serveur proxy SOCKS est utilisé. La valeur 1 signifie que toutes les requêtes pour le protocole spécifié transitent par un serveur SOCKS. La valeur 0 (zéro) signifie que les requêtes ne transitent pas par un quelconque serveur SOCKS.

Le paramètre *nomServeurProxy* retourne le nom ou l'adresse IP du serveur proxy SOCKS utilisé.

Le paramètre *portProxy* retourne le numéro du port utilisé avec le protocole spécifié pour communiquer avec le serveur proxy SOCKS.

Le paramètre *idUtilisateurProxy* retourne l'ID de l'utilisateur.

IT_GetTimeOut

IT_GetTimeOut (timeOut) -> Résultat

Paramètre	Type		Description
timeOut	Entier	⇦	Délai d'attente en secondes
Résultat	Entier	⇨	Code d'erreur

Description

La commande *IT_GetTimeOut* renvoie la valeur courante du délai d'attente (timeout) pour les commandes énumérées dans la description de la commande *IT_SetTimeOut*.

Le paramètre *timeOut* retourne la valeur courante du délai d'attente en secondes.

IT_MyTCPAddr

IT_MyTCPAddr (adresse_IP ; sousRéseau) -> Résultat

Paramètre	Type		Description
adresse_IP	Chaîne	⇐	Adresse IP de la machine de l'utilisateur
sousRéseau	Chaîne	⇐	Masque de sous-réseau au format IP
Résultat	Entier	⇒	Code d'erreur

Description

La commande *IT_MyTCPAddr* renvoie l'adresse IP de la machine qui exécute la commande.

Le paramètre *adresse_IP* retourne une chaîne de caractères contenant l'adresse IP.

Le paramètre *sousRéseau* retourne une chaîne de caractères contenant le masque de sous-réseau de l'adresse IP.

IT_Platform

IT_Platform -> Résultat

Paramètre	Type		Description
Résultat	Entier		Type de plate-forme (1 = Mac OS, 2 = Windows)

Description

La fonction *IT_Platform* retourne un entier indiquant la plate-forme en cours d'exploitation.
La fonction retourne 1 pour Mac OS et 2 pour Windows.

Exemple

```
C_BOOLEAN(◇ITnative)
◇ITnative:=(IT_Platform=1)
```

IT_PPPConnect

IT_PPPConnect (profilPPP) -> Résultat

Paramètre	Type		Description
profilPPP	Chaîne	→	Nom de connexion à distance (chaîne vide sous Mac OS, nom sous Windows)
Résultat	Entier	←	Code d'erreur

Description

La commande *IT_PPPConnect* ouvre une connexion à distance sous Mac OS ou la connexion à distance désignée via le paramètre *profilPPP* sous Windows. La commande retourne un code d'erreur si la connexion ne peut pas être ouverte. Une connexion à distance (ou "connexion commutée") est établie via une ligne téléphonique (modem ou ligne spécialisée).

Cette commande doit être appelée chaque fois que vous voulez exécuter un ensemble de commandes 4DIC fonctionnant en ligne. A l'issue de cette session, vous devez exécuter *IT_PPPDisconnect* afin de refermer la connexion courante.

Le protocole PPP (Point-to-Point Protocol) est utilisé pour la communication entre deux ordinateurs à l'aide d'une interface série, typiquement un PC connecté à un serveur via une ligne téléphonique. Par exemple, votre fournisseur d'accès Internet peut vous proposer une connexion PPP afin que son serveur réponde à vos requêtes, les transmette sur Internet et vous retourne les réponses. Schématiquement, le protocole encapsule les paquets TCP/IP envoyés par votre machine et les transmet au serveur qui pourra les diffuser sur Internet.

Le protocole PPP est généralement préférable au précédent standard "de facto" SLIP (Serial Line Internet Protocol) car il permet de gérer des communications synchrones et asynchrones. PPP peut en outre partager une ligne téléphonique avec d'autres utilisateurs et dispose d'une fonction de détection d'erreurs, avantages dont est dépourvu le protocole SLIP. Si vous avez le choix, préférez PPP.

IT_PPPDisconnect

IT_PPPDisconnect (profilPPP) -> Résultat

Paramètre	Type		Description
profilPPP	Chaîne	➡	Nom de connexion à distance (chaîne vide sous Mac OS, nom optionnel sous Windows)
Résultat	Entier	⮌	Code d'erreur

Description

La commande *IT_PPPDisconnect* referme la connexion à distance courante préalablement ouverte à l'aide de la commande *IT_PPPConnect*.

Le paramètre *profilPPP* désigne la connexion à refermer.
Sous Windows, ce paramètre est utile lorsque plusieurs connexions PPP sont ouvertes simultanément. L'utilisation de ce paramètre garantit l'exécution de la commande quelle que soit la configuration réseau de l'utilisateur.

Sous Windows

- Si une seule connexion PPP est ouverte et si le paramètre *profilPPP* est omis ou contient une chaîne vide, *IT_PPPDisconnect* referme la connexion ouverte.
- Si plusieurs connexions PPP sont ouvertes et si le paramètre *profilPPP* est omis ou contient une chaîne vide, *IT_PPPDisconnect* retourne une erreur ; aucune connexion n'est refermée.
- Si *profilPPP* est passé et est valide, la connexion spécifiée est refermée, quel que soit le nombre de connexions ouvertes.

Sous Mac OS

Le paramètre *profilPPP* est ignoré.

IT_PPPStatus

IT_PPPStatus (profilPPP) -> Résultat

Paramètre	Type		Description
profilPPP	Chaîne	➡	Nom de connexion à distance (chaîne vide sous Mac OS, nom optionnel sous Windows)
Résultat	Entier	➲	1 = connecté, 0 = en cours de connexion, -1 = erreur

Description

La commande *IT_PPPStatus* vous permet de tester le statut d'une connexion à distance ouverte à l'aide de la commande *IT_PPPConnect* ou manuellement.

Le paramètre *profilPPP* désigne la connexion à tester. Sous Windows, ce paramètre est optionnel mais peut être utile pour garantir l'exécution de la commande quelle que soit la configuration réseau de l'utilisateur.

Sous Windows

- Si *profilPPP* est passé et est valide, le statut de la connexion spécifiée est retourné.
- Si *profilPPP* est omis ou contient une chaîne vide, *IT_PPPStatus* retourne :
  - si plusieurs connexions sont ouvertes, -1
  - si une seule connexion est ouverte, le statut de la connexion.

Sous Mac OS

Le paramètre *profilPPP* est ignoré.
IT_PPPStatus retourne un entier indiquant le statut de la connexion :

- 1 si la connexion est établie,
- 0 si la connexion est en cours d'établissement,
- -1 si la connexion a échoué ou s'il n'y a aucune connexion.

Exemple

```
 //Méthode GetMessages (méthode exécutée dans un process)
If(mPPPConnect($vPPPProfil;120))
   $vErrCode:=IT_MacTCPInit
   If($vErrCode=0)
      $vErrCode:=POP3_Login...
      ...
   Else
      ALERT("Connection failed")
   End if
End if

 //Méthode mPPPConnect
C_BOOLEAN($0) //Vrai si on est déjà connecté, Faux si la connexion a échoué
C_TEXT($1) //Chaîne vide sous Mac OS, Nom sous Windows
C_LONGINT($2) //Timeout en secondes

If(IT_PPPStatus=1)
   $0:=True //On est déjà connecté
Else
   $vTimeoutLength:=$2
   $vTimeout:=False
   $vErr:=IT_PPPConnect($1)
   If($vErr=0)
```

```
        $vStart:=Current time
        Repeat
            DELAY PROCESS(Current process;30)
            $vStatus:=IT_PPPStatus($1)
            $vTimeout:=((Current time-$vStart)>$vTimeoutLength)
        Until(($vStatus=1)|$vTimeout)  //Connexion ou timeout
        If(Not($vTimeout))
            $0:=True  //Connexion
        End if
    End if  //··· $Err = 0
End if
```

IT_SetPort

IT_SetPort (protocole ; port) -> Résultat

Paramètre	Type		Description
protocole	Entier	⇒	1 = FTP ; 2 = SMTP ; 3 = POP3 ; 4 = IMAP ; 12 = SMTP TLS ; 13 = POP3 TLS ; 14 = IMAP TLS
port	Entier	⇒	Numéro de port
Résultat	Entier	⇐	Code d'erreur

Description

Pour le *protocole* spécifié, la commande *IT_SetPort* dirige toutes les communications ultérieures utilisant ce protocole vers le *port* spécifié.

IT_SetProxy

IT_SetProxy (protocole ; typeProxy ; nomServeurProxy ; portProxy ; idUtilisateurProxy) -> Résultat

Paramètre	Type		Description
protocole	Entier	➡	1 = FTP ; 2 = SMTP ; 3 = POP3 ; 4 = IMAP
typeProxy	Entier	➡	0 = Aucun ; 1 = SOCKS
nomServeurProxy	Chaîne	➡	Nom ou adresse IP du serveur proxy SOCKS
portProxy	Entier	➡	Port du proxy auquel se connecter
idUtilisateurProxy	Texte	➡	ID d'utilisateur pour SOCKS
Résultat	Entier	↩	Code d'erreur

Description

La commande *IT_SetProxy* permet d'ouvrir une connexion au moyen du protocole spécifié, puis d'envoyer toutes les requêtes proposées au protocole spécifié via le serveur SOCKS (Proxy SOCKS). Si vous vous connectez uniquement en Intranet, vous n'aurez généralement pas à passer par le serveur SOCKS, sauf si le paramétrage du firewall ("pare feu") de votre réseau le requiert. La commande *IT_SetProxy* a une portée interprocess et s'applique à toutes les sessions de communication utilisant le protocole spécifié, quel que soit le process 4D.

Note : Socks (ou "SOCKS") est un protocole utilisable par un serveur proxy. Il permet de filtrer les requêtes des utilisateurs du réseau d'une entreprise, lorsque celles-ci doivent être transmises sur Internet. Si votre poste de travail se trouve derrière un firewall et si vous souhaitez accéder à des informations sur Internet, le serveur SOCKS reçoit votre requête, la transmet à travers le firewall, puis récupère et renvoie les informations à votre application cliente.

Le paramètre *protocole* définit le protocole qui doit être filtré par le serveur proxy SOCKS spécifié. Passez 1 pour désigner le protocole FTP, 2 pour le protocole SMTP, 3 pour le protocole POP3 et 4 pour le protocole IMAP.

Le paramètre *typeProxy* indique si le protocole spécifié doit ou non être filtré par un serveur proxy SOCKS. Passez 1 pour faire transiter toutes les requêtes du protocole spécifié par le serveur SOCKS spécifié, sinon passez 0.

Le paramètre *nomServeurProxy* contient le nom ou l'adresse IP du serveur proxy SOCKS.

Le paramètre *portProxy* contient le port à utiliser pour que le protocole spécifié communique avec le serveur proxy SOCKS.

Le paramètre *idUtilisateurProxy* identifie l'utilisateur. L'ID d'utilisateur est attribué par votre administrateur réseau. *idUtilisateurProxy* peut être un texte vide ("").

Exemple

La méthode suivante permet d'acheminer toutes les connexions FTP via le serveur proxy SOCKS spécifié.

```
$err:=IT_SetProxy(1;1;$ajouterProxy;$PortProxy;"")  `Proxy SOCKS FTP
$err:=FTP_Login("ftp.4d.com";"anonymous";"dbody@aol.com";$ftpID)
$err:=FTP_GetFileInfo($ftpID;$vchemin;$vtaille;$vdateModif)
$err:=FTP_Receive($ftpID;$vchemin;"";0)
$err:=FTP_Logout($ftpID)
```

Note : Par souci de clarté, cet exemple ne contient pas de vérification d'erreur.

L'instruction suivante permet de ne plus passer par le serveur proxy SOCKS lors des requêtes FTP.

```
$err:=IT_SetProxy(1;0;$ajouterProxy;$portProxy;"")
```

IT_SetTimeOut

IT_SetTimeOut (timeOut) -> Résultat

Paramètre	Type		Description
timeOut	Entier	➡	Délai d'attente en secondes (0 à 127)
Résultat	Entier	⮌	Code d'erreur

Description

La commande *IT_SetTimeOut* vous permet de définir la valeur du délai d'attente maximal (Timeout) alloué par défaut pour l'exécution des commandes Internet de 4D. Cette valeur doit être exprimée en secondes et être comprise entre 0 et 127. Par défaut, la valeur de timeout est de 30 secondes.

Passez dans *timeOut* la valeur souhaitée. Les commandes suivantes sont affectées par *IT_SetTimeOut* :

TCP_Open
FTP_Login
FTP_Append
FTP_Send
FTP_Receive
SMTP_QuickSend
SMTP_Send
POP3_Login
POP3_BoxInfo
POP3_Delete
POP3_Reset
POP3_MsgInfo
POP3_MsgLstInfo
POP3_GetMessage
POP3_MsgLst
POP3_DownLoad
POP3_VerifyID
POP3_UIDToNum
IMAP_Login
IMAP_VerifyID
IMAP_Capability
IMAP_ListMBs
IMAP_SubscribeMB
IMAP_GetMBStatus
IMAP_SetCurrentMB
IMAP_Delete
IMAP_MsgInfo
IMAP_MsgLstInfo
IMAP_GetMessage
IMAP_MsgLst
IMAP_SetFlags
IMAP_GetFlags
IMAP_Search
IMAP_MsgFetch
IMAP_Download
IMAP_CopyToMB
IMAP_CreateMB
IMAP_RenameMB
IMAP_DeleteMB
NET_Finger
NET_Ping
NET_Time

Note : La définition du *timeOut* à 0 (zéro) pour la commande *TCP_Listen* lui permet d'être à l'écoute indéfiniment. Veillez à passer une autre valeur après avoir utilisé cette commande. De plus, cette valeur est également utilisée pour les "timeouts TCP/IP" ET le "timeout d'attente de réponse". Si vous fixez ce délai d'attente à zéro, aucune réponse n'aura le temps d'être reçue.

IT_TCPversion

IT_TCPversion (typePile ; versionPile) -> Résultat

Paramètre	Type		Description
typePile	Entier	⇐	0 = Aucun, 3 = WinSock, 4 = BSD
versionPile	Texte	⇐	Numéro de version de la pile TCP
Résultat	Entier	⇒	Code d'erreur

Description

La commande *IT_TCPversion* retourne des informations sur le type de pile TCP en cours d'utilisation avec les commandes Internet de 4D. Le type de pile varie selon la plate-forme. Sur Macintosh, la pile TCP BSD est prise en charge. Sous Windows, la pile TCP WinSock est prise en charge.

Le paramètre *typePile* retourne un code exprimant le type de pile TCP utilisé, en fonction du tableau suivant :

Code	Pile TCP
0	Aucun
1	MacTCP (obsolète, cf. Notes de compatibilité)
2	Open Transport (obsolète, cf. Notes de compatibilité)
3	WinSock
4	BSD Sockets

Notes de compatibilité :

- A compter de la version 6.8 de 4D Internet Commands, MacTCP n’est plus pris en charge. Par conséquent, le paramètre *typePile* ne retourne plus la valeur 1.
- A compter de la version 2004 de 4D Internet Commands, Open Transport n’est plus pris en charge. Par conséquent, le paramètre *typePile* ne retourne plus la valeur 2.

Le paramètre *versionPile* retourne sous forme de texte le numéro de version de la pile TCP utilisée.

IT_Version

IT_Version -> Résultat

Paramètre	Type		Description
Résultat	Chaîne		Chaîne de la version

Description

La fonction *IT_Version* renvoie une chaîne de caractères indiquant le numéro de version du plug-in 4D Internet Commands.

Exemple

L'exemple suivant affiche une boîte de dialogue d'alerte indiquant la version de 4D Internet Commands utilisée :

```
ALERT("4D Internet Commands, version : "+IT_Version)
```

IT_MacTCPInit

IT_MacTCPInit -> Résultat

Paramètre	Type		Description
Résultat	Entier		Code d'erreur

Note de compatibilité

La commande *IT_MacTCPInit* est désormais sans effet, elle ne doit plus être utilisée.

IT_MacTCPVer

IT_MacTCPVer (codeVersion) -> Résultat

Paramètre	Type		Description
codeVersion	Entier	⇐	Code de la version de MacTCP installée
Résultat	Entier	⇒	Code d'erreur

Note de compatibilité

La commande *IT_MacTCPVer* est devenue obsolète avec l'ajout de la commande *IT_TCPversion* qui permet d'obtenir des informations plus complètes sur Open Transport et Winsock.

Description

A compter de la version 6.8 de 4D Internet Commands, MacTCP n’est plus pris en charge. Par conséquent, le paramètre *codeVersion* retourne toujours 0, quels que soient la plate-forme et le système d’exploitation.

Annexes

Annexe A, Conseils de programmation

Exécution des commandes dans des boucles 'Au cas ou'

Dans de nombreux exemples de ce manuel, on utilise une structure de programmation particulière. Ces exemples exécutent des séries de commandes en utilisant l'instruction **Au cas ou** de manière peu orthodoxe.

En fait, de nombreuses fonctionnalités des commandes Internet de 4D Internet requièrent l'exécution complète d'une séquence de commandes. L'échec d'une seule commande de la séquence étant suffisant pour interrompre la poursuite du processus, l'utilisation de **Si** imbriqués sur de nombreux niveaux serait fastidieux :

```
If(SMTP_New($smtp_id)=0)
   If(SMTP_Host($smtp_id;◇pref_Serveur)=0)
      If(SMTP_From($smtp_id;vDe)=0)
         If(SMTP_To($smtp_id;vA)=0)
            etc, etc
         End if
      End if
   End if
End if
```

La solution employée dans ce manuel consiste à s'appuyer sur la manière dont 4D exécute les instructions **Au cas ou**. Chaque condition d'une instruction **Au cas ou** est exécutée par 4D pour déterminer si la valeur renvoyée est **Vraie** ou **Fausse**. Lorsque chaque condition d'une instruction **Au cas ou** renvoie une valeur fausse, toutes les conditions sont donc exécutées. Les lignes suivantes remplacent le code décrit plus haut :

```
$EnvoyeOK:=False  `Ce "drapeau" indique si tous les appels ont été transmis
Case of
   :(SMTP_New($smtp_id)#0)
   :(SMTP_Host($smtp_id;◇pref_Serveer)#0)
   :(SMTP_From($smtp_id;vDe)#0)
   :(SMTP_To($smtp_id;vA)#0)
   :(SMTP_Subject($smtp_id;vSujet)#0)
   :(SMTP_Body($smtp_id;vMessage)#0)
   :(SMTP_Send($smtp_id)#0)
   Else
      $EnvoyeOK:=True  `Le message a été composé et envoyé
End case
If($smtp_id#0)  `Si un message a été créé en mémoire, nous devons l'effacer maintenant
   $OK:=SMTP_Clear($smtp_id)
End if
```

Dans cet exemple, chaque commande renvoie l'erreur 0 (zéro) si son exécution s'est déroulée correctement. Pour pouvoir évaluer chaque condition, 4D doit en fait exécuter chaque ligne. Comme chaque condition compare le résultat à "différent de zéro", la valeur renvoyée est toujours **Fausse** et 4D ne trouve pas de condition à laquelle s'arrêter jusqu'à ce que l'une des commandes échoue. Si chaque commande est exécutée correctement, 4D poursuit l'exécution de la méthode jusqu'à la condition **Sinon** dans laquelle le drapeau $EnvoyeOK prend la valeur **Vraie** afin d'indiquer que le message a été composé et envoyé sans incident.

Recommandations pour une réponse automatique à un courrier POP3 ou IMAP

Si vous envisagez de mettre en place, à l'intérieur de votre base de données, un système de messagerie permettant notamment aux utilisateurs de "répondre" aux courriers qu'ils ont reçu, voici quelques recommandations tirées de la RFC 822 :

- L'adresse électronique de l'"Expéditeur" (Sender) ne doit recevoir que des réponses relatives à des problèmes de distribution de courrier et **non** des réponses se rapportant au sujet traité dans le message. En l'absence d'en-tête "Expéditeur" (Sender), les problèmes doivent être envoyés à l'adresse figurant dans l'en-tête "Emetteur(s)" (From).

- L'adresse de l'"Expéditeur" (Sender) ne doit jamais être utilisée avec des automates. L'automate devra plutôt utiliser l'en-tête "Réponse à" (Reply-To) ou "Emetteur(s)" (From) pour répondre à un message, le choix de l'en-tête le plus approprié dépendant des facteurs indiqués ci-dessous.

- Si l'en-tête "Réponse à" (Reply-To) existe et contient une ou plusieurs adresses, les réponses doivent alors être adressées aux personnes de cette liste. Les adresses de l'en-tête "Réponse à" (Reply-To) ont priorité sur celles figurant dans l'en-tête "Emetteur(s)" (From). En l'absence d'en-tête "Réponse à" mais en présence d'un en-tête "Emetteur(s)", les réponses doivent être envoyées aux adresses indiquées dans ce dernier.

Ces recommandations ont pour seul but de faciliter la prise de décision lorsque l'adressage du courrier est géré par programmation dans le cas d'actions de type "Réponse". Une fois le message de réponse créé, l'utilisateur final peut parfaitement annuler ces paramétrages par défaut avant d'envoyer le message.

Annexe B, Numéros des ports TCP

Choix du numéro de port

- 0 à 1023 (Ports réservés) : Ces ports sont affectés par l'I.A.N.A. (Internet Assigned Numbers Authority) et sur la plupart des systèmes ne peuvent être utilisés que par des process système (ou racine) ou par des programmes exécutés par des utilisateurs disposant de privilèges d'accès avancés.
  - 20 et 21 FTP;
  - 23 TELNET;
  - 25 SMTP;
  - 37 NTP;
  - 80 et 8080 HTTP;
  - 443 HTTPS.
- 1024 à 49151 (Ports enregistrés) : Ces ports sont enregistrés par l'I.A.N.A. et peuvent être utilisés sur la plupart des systèmes par des process utilisateurs ou par des programmes exécutés par des utilisateurs sans privilèges particuliers (routeurs, applications spécifiques...)
- 49152 à 65535 (Ports dynamiques et/ou privés) : Ces ports sont d'utilisation libre.

Les personnes souhaitant utiliser les commandes TCP/IP pour synchroniser des bases de données doivent utiliser des numéros de port supérieurs à 49151.

Pour de plus amples informations, veuillez visiter le site Web de l'I.A.N.A. : http://www.iana.org

Numéros de port TCP

daytime	13	Daytime
qotd	17	Quote of the Day
ftp-data	20	File Transfer [données par défaut]
ftp	21	File Transfer [contrôle]
telnet	23	Telnet
smtp	25	Simple Mail Transfer
time	37	Time
nicname	43	Who Is
domain	53	Domain Name Server
sql*net	66	Oracle SQL*NET
gopher	70	Gopher
finger	79	Finger
http	80	World Wide Web HTTP
poppassd	106	Password Server
rtelnet	107	Remote Telnet Service
pop2	109	Post Office Protocol - Version 2
pop3	110	Post Office Protocol - Version 3
sunrpc	111	SUN Remote Procedure Call
auth	113	Authentication Service
sftp	115	Simple File Transfer Protocol
sqlserv	118	SQL Services
nntp	119	Network News Transfer Protocol
ntp	123	Network Time Protocol
pwdgen	129	Password Generator Protocol
imap2	143	Interactive Mail Access Protocol v2
news	144	NewS
sql-net	150	SQL-NET
multiplex	171	Network Innovations Multiplex
cl/1	172	Network Innovations CL/1
at-rtmp	201	AppleTalk Routing Maintenance
at-nbp	202	AppleTalk Name Binding
at-3	203	AppleTalk Unused
at-echo	204	AppleTalk Echo
at-5	205	AppleTalk Unused
at-zis	206	AppleTalk Zone Information
at-7	207	AppleTalk Unused
at-8	208	AppleTalk Unused
ipx	213	IPX
netware-ip	396	Novell Netware sur IP
timbuktu	407	Timbuktu
https	443	Protocole sécurisé
conference	531	conversation
netnews	532	readnews
netwall	533	Pour émission d'urgence
uucp	540	uucpd
uucp-rlogin	541	uucp-rlogin
whoami	565	whoami
ipcserver	600	Sun IPC server
phonebook	767	téléphone
accessbuilder	888	AccessBuilder

Annexe C, Codes d'erreurs de 4D Internet Commands

Toutes les commandes Internet de 4D (à l'exception de *IT_ErrorText* et *IT_Version*) sont des fonctions retournant un entier comme résultat. Cet entier contient un numéro d'erreur que la commande doit retourner à la base de données 4D. Si une commande aboutit, un zéro est renvoyé.

L'origine d'une erreur peut généralement être déterminée à partir de l'intervalle de valeurs dans lequel se situe son numéro. Le tableau suivant vous permet de distinguer les sources d'erreurs en fonction de leur numéro :

Numéro d'erreur	Générée par
Erreur < 0	Système d'exploitation ou couche réseau WinSock
0	Pas d'erreur
Erreur 1 -> 61	Couche réseau BSD
Erreur >= 10000	Commandes Internet de 4D

Codes d'erreurs de 4D Internet Commands

Si une erreur se produit durant l'exécution d'une commande Internet de 4D, une des valeurs suivantes sera retournée :

10000 user cancelled a dialog or progress.
10001 unimplemented Internet command.
10002 invalid array type.
10003 no more (TCP,SMTP,POP3, etc.) references available.
10004 invalid reference.
10005 need a "Host" for use in the "SMTP_Send" command.
10006 need a "From" for use in the "SMTP_Send" command.
10007 need a recipient for use in the "SMTP_Send" command.
10008 already logged in.
10009 error trying to make a POP3 connection.
10010 error with POP3 USER.
10011 error with POP3 PASS.
10012 error with POP3 QUIT.
10013 error with POP3 STAT.
10014 error with POP3 LIST.
10015 error with POP3 UIDL.
10016 error with POP3 DELE.
10017 error with POP3 RSET.
10018 invalid message number.
10019 invalid character offset.
10020 invalid character length.
10021 error with POP3 RETR.
10022 field was not found in mail Header.
10023 no attachments found.
10024 error in processing BinHex.
10025 BinHex checksum error.
10026 Internet commands unavailable. Probably because MacTCP is not installed
10027 Connection no longer exists
10028 Exceeded 32k limit
10029 Error with POP3 NOOP
10030 POP3 session was closed by the server
10031 Error with POP3 APOP
10032 Unknown or invalid response.
10033 SMTP 421 - Service not available, closing transmission channel.
10034 SMTP 450 - Requested mail action not taken: mailbox unavailable.
10035 SMTP 451 - Requested action aborted: local error in processing.
10036 SMTP 452 - Requested action not taken: insufficient system storage.
10037 SMTP 500 - Syntax error, command unrecognized.
10038 SMTP 501 - Syntax error in parameters or arguments.
10039 SMTP 502 - Command not implemented.
10040 SMTP 503 - Bad sequence of commands.
10041 SMTP 504 - Command parameter not implemented.
10042 SMTP 550 - Requested action not taken: mailbox unavailable.
10043 SMTP 551 - User not local; please try <forward-path>.
10044 SMTP 552 - Requested mail action aborted: exceeded storage allocation.
10045 SMTP 553 - Requested action not taken: mailbox name not allowed.
10046 SMTP 554 - Transaction failed.
10047 FTP 421 - Service not available, closing control connection.
10048 FTP 425 - Can't open data connection.
10049 FTP 426 - Connection closed; transfer aborted.
10050 FTP 450 - Requested file action not taken. File unavailable (e.g.,file busy).
10051 FTP 451 - Requested action aborted: local error in processing.
10052 FTP 452 - Requested action not taken. Insufficient storage space in system.
10053 FTP 500 - Syntax error, command unrecognized.
10054 FTP 501 - Syntax error in parameters or arguments.
10055 FTP 502 - Command not implemented.
10056 FTP 503 - Bad sequence of commands.

10057	FTP 504 - Command not implemented for that parameter.
10058	FTP 530 - Not logged in.
10059	FTP 532 - Need account for storing files.
10060	FTP 550 - Requested action not taken. File unavailable (e.g., file not found, no access).
10061	FTP 551 - Requested action aborted: page type unknown.
10062	FTP 552 - Requested file action aborted. Exceeded storage allocation (for current directory or dataset).
10063	FTP 553 - Requested action not taken. File name not allowed.
10064	No response has been received within the given timeout period.
10065	Not a FTP file.
10066	Error in processing Base64.
10067	Error in processing AppleSingle.
10068	Error in processing Quoted-Printable.
10069	FTP session was closed by the server.
10070	Not a FTP directory.
10071	TCP session was closed by the server
10072	Invalid encode kind
10073	Invalid decode kind
10074	An asynchronous DNR call did not complete
10075	An asynchronous OpenTransport call did not complete
10076	OpenTransport bind failed
10077	OpenTransport connect failed
10078	Maximum MacTCP streams reached
10079	Error in processing uuencode
10080	Cannot load ICMP library
10081	Error in processing MacBinary
10082	MacBinary checksum error
10083	Could not open a file
10084	No FTP information received
10085	Unknown FTP information received
10086	Proxy connection failed
10087	Standard file I/O error
10088	FTP reentrant error
10089	SLI.DLL is not loaded
10091	Error trying to make an IMAP connection
10092	A maibox is not selected
10093	Invalid message part
10094	Error with IMAP LOGIN
10095	Error with IMAP LOGOUT
10096	Error with IMAP CAPABILITY
10097	Error with IMAP SELECT
10098	Error with IMAP FETCH
10099	Error with IMAP PARTIAL
10100	Error with IMAP STORE
10101	Error with IMAP EXPUNGE
10102	Error with IMAP SEARCH
10103	Error with IMAP COPY
10104	Error with IMAP CREATE
10105	Error with IMAP DELETE
10106	Error with IMAP RENAME
10107	Error with IMAP SUBSCRIBE
10108	Error with IMAP UNSUBSCRIBE
10109	Error with IMAP LIST
10110	Error with IMAP LSUB
10111	Error with IMAP STATUS
10112	Error with IMAP CLOSE
10113	Error with AUTHENTICATION

Codes d'erreurs BSD

1	Operation not permitted
4	Interrupted system call
13	Permission denied
14	Bad address
22	Invalid argument
24	Too many open files
35	Operation would block
36	Operation now in progress
37	Operation already in progress
38	Socket operation on non-socket
39	Destination address required
40	Message too long
41	Protocol wrong type for socket
42	Protocol not available
43	Protocol not supported
44	Socket type not supported
45	Operation not supported
46	Protocol family not supported
47	Address family not supported by protocol family
48	Address already in use
49	Can't assign requested address
50	Network is down
51	Network is unreachable
52	Network dropped connection on reset
53	Software caused connection abort
54	Connection reset by peer
55	No buffer space available
56	Socket is already connected
57	Socket is not connected
58	Can't send after socket shutdown
60	Operation timed out
61	Connection refused

Codes d'erreurs WinSock

-10004	Blocking call cancelled
-10013	Permission denied
-10014	Bad address
-10022	Invalid argument
-10024	No more sockets available
-10035	Non-blocking socket would block
-10036	Illegal WinSock function invoked while a blocking function is in progress
-10037	An attempt was made to cancel an asynchronous operation that has already completed
-10038	Specified socket descriptor is not valid for this application
-10039	Destination address was required but none was supplied to the function
-10040	Datagram too large for buffer
-10041	Specified protocol does not match the other parameters in the call
-10042	Protocol option is unknown or invalid
-10043	Specified protocol is not supported by the Windows Sockects implementation
-10044	Specified socket type is not supported by the specified address family
-10045	Socket does not support the specified operation
-10046	Protocol family not supported
-10047	Specified address family is not supported by the Windows Sockects implementation or cannot be used with the indicated socket
-10048	Specified address is already in use
-10049	Specified address is not available from the local machine
-10050	Problem with the network subsystem
-10051	Network cannot be reached from this host at this time
-10052	Connection was dropped and must be reset
-10053	Connection was aborted because of a timeout or other error condition
-10054	Connection was reset by the remote host
-10055	Windows Sockets implementation is out of buffer space or the space provided in an API call by the application was too small to hold the requested information
-10056	Specified socket is already connected
-10057	Specified socket is not connected
-10058	Socket has had the requested functionality shut down
-10060	Connection attempt timed out before the connection could be established
-10061	Connection attempt was forcefully rejected
-10091	Network subsystem is not yet ready for communication
-10092	Windows Sockets DLL does not support the requested Winsock protocol version
-10093	Windows Sockets not initialized
-11001	Requested database information does not exist; as confirmed by an authoritative host
-11002	Requested information was not found but the answer was not authoritative
-11003	Non-recoverable error occurred
-11004	Name supplied was valid but no information of the requested type is in the database

Codification SMTP

Les valeurs suivantes **ne sont pas** des codes d'erreurs retournés par les commandes Internet de 4D. Ce sont les codes de réponses définis pour le protocole SMTP afin d'indiquer l'état courant de la communication en mode client-serveur. Cette liste est particulièrement utile dans le cas où vous souhaitez créer votre propre système de messagerie à l'aide des commandes TCP de bas niveau.

211	System status, or system help reply
214	Help message [Information on how to use the receiver or the meaning of a particular non-standard command; this reply is useful only to the human user]
220	<domain> Service ready
221	<domain> Service closing transmission channel
250	Requested mail action okay, completed
251	User not local; will forward to <forward-path>
354	Start mail input; end with <CRLF>.<CRLF>
421	<domain> Service not available, closing transmission channel [This may be a reply to any command if the service knows it must shut down]
450	Requested mail action not taken: mailbox unavailable [E.g., mailbox busy]
451	Requested action aborted: local error in processing
452	Requested action not taken: insufficient system storage
500	Syntax error, command unrecognized [This may include errors such as command line too long]
501	Syntax error in parameters or arguments
502	Command not implemented
503	Bad sequence of commands
504	Command parameter not implemented
550	Requested action not taken: mailbox unavailable [E.g., mailbox not found, no access]
551	User not local; please try <forward-path>
552	Requested mail action aborted: exceeded storage allocation
553	Requested action not taken: mailbox name not allowed [E.g., mailbox syntax incorrect]
554	Transaction failed

Codification FTP

Les valeurs suivantes **ne sont pas** des codes d'erreurs retournés par les commandes Internet de 4D. Ce sont les codes de réponses définis pour le protocole FTP afin d'indiquer l'état courant de la communication en mode client-serveur. Cette liste est particulièrement utile dans le cas où vous souhaitez créer votre propre système de transfert de fichiers à l'aide des commandes TCP de bas niveau.

110 Restart marker reply. In this case, the text is exact and not left to the particular implementation; it must read:

MARK yyyy = mmmm

Where yyyy is User-process data stream marker, and mmmm server's equivalent marker (note the spaces between markers and "=").

120 Service ready in nnn minutes.

125 Data connection already open; transfer starting.

150 File status okay; about to open data connection.

200 Command okay.

202 Command not implemented, superfluous at this site.

211 System status, or system help reply.

212 Directory status.

213 File status.

214 Help message on how to use the server or the meaning of a particular non-standard command. This reply is useful only to the human user.

215 NAME system type. Where NAME is an official system name from the list in the Assigned Numbers document.

220 Service ready for new user.

221 Service closing control connection. Logged out if appropriate.

225 Data connection open; no transfer in progress.

226 Closing data connection. Requested file action successful (for example, file transfer or file abort).

227 Entering Passive Mode (h1,h2,h3,h4,p1,p2).

230 User logged in, proceed.

250 Requested file action okay, completed.

257 "PATHNAME" created.

331 User name okay, need password.

332 Need account for login.

350 Requested file action pending further information.

421 Service not available, closing control connection. This may be a reply to any command if the service knows it must shut down.

425 Can't open data connection.

426 Connection closed; transfer aborted.

450 Requested file action not taken. File unavailable (file busy).

451 Requested action aborted: local error in processing.

452 Requested action not taken. Insufficient storage space in system.

500 Syntax error, command unrecognized. This may include errors such as command line too long.

501 Syntax error in parameters or arguments.

502 Command not implemented.

503 Bad sequence of commands.

504 Command not implemented for that parameter.

530 Not logged in.

532 Need account for storing files.

550 Requested action not taken. File unavailable (e.g., file not found, no access).

551 Requested action aborted: page type unknown.

552 Requested file action aborted. Exceeded storage allocation (for current directory or dataset).

553 Requested action not taken. File name not allowed.

Annexe D, Informations supplémentaires...

Les références suivantes sont les URLs complémentaires relatifs aux protocoles Internet.

http://www.internic.net : Pour comprendre ce qu'est un nom de domaine et connaître la démarche pour en enregistrer un.

http://www.ietf.org : Site de l'Internet Engineering Task Force (IETF).

http://www.rfc-editor.org : Pour comprendre ce qu'est une RFC et savoir comment les rechercher (voir aussi *http://www.rfc-editor.org/rfc.html*).

ftp://ftp.isi.edu/in-notes/rfc821.txt : Simple Mail Transfer Protocol -- RFC 821.

http://www.w3c.org : Tout ce qu'il faut savoir sur le World Wide Web.

http://www.imap.org : Site entièrement dédié au protocole IMAP.

Annexe E, simuler l'envoi de mail dans un fichier local

Tester et déboguer du code qui envoie des mails peut être très difficile ; lorsqu'un courrier électronique n'est pas reçu correctement, la cause peut être multiple : le réseau, le fournisseur d'accès, le logiciel client, etc.

Pour vous aider, nous avons ajouté la possibilité d'envoyer les emails dans un fichier local au lieu de les envoyer aux destinataires. Grâce à cela, il vous suffit de modifier le fichier en un fichier EML - ce qui est très simple - et ce fichier pourra ainsi afficher les résultats dans MS Outlook. Vous pouvez également inclure des fichiers de courrier électronique dans des procédures de test unitaire.

Code pour un test en local

Vous pouvez lancer le code en local :

```
$err:=SMTP_SetPrefs(0;15;0) // Corps : UTF-8 & QuotedPrintable, En-tête : UTF-8 & Base64
$err:=SMTP_Charset(1;1) // Applique les préférences au Corps et à l'En-tête (Corps : UTF-8 & QuotedPrintable & En-tête : UTF-8 & Base64)

 //$hostName:="smtp.gmail.com" // Envoi à travers le réseau en utilisant smtp.gmail.com
$hostName:="file:C:\\Users\\MyWinUser\\Desktop\\test.txt" // ou envoi dans un fichier texte
$msgTo:="mail.to@gmail.com"
$msgFrom:="mail.sender@gmail.com"

$mailSubject:="テストメール(v17 4372) " //test en utilisant des caractères étendus
$mailBody:="日本語で終わる"
$err:=SMTP_QuickSend($hostName;$msgFrom;$msgTo;$mailSubject;$mailBody;0;0;$msgFrom;"password")
```

Et vous obtiendrez le fichier texte suivant :

```
<mail.sender@gmail.com>
<mail.to@gmail.com>

Mime-Version: 1.0
Content-Type: text/plain;charset="utf-8"
Content-Transfer-Encoding: quoted-printable
Date: Fri, 08 Jul 2016 16:45:24 +0200
To: mail.to@gmail.com
From: mail.sender@gmail.com
Subject: =?utf-8?B?44OG44K544OI44Oh44O844Or77yIdjE3IDQzNzIpIA==?=

=E6=97=A5=E6=9C=AC=E8=AA=9E=E3=81=A7=E7=B5=82=E3=82=8F=E3=82=8B
```

Si vous souhaitez ouvrir ce fichier en tant qu'email standard MS Outlook :

1. Supprimez toutes les lignes avant "Mime-Version: 1.0":

```
Mime-Version: 1.0
Content-Type: text/plain;charset="utf-8"
Content-Transfer-Encoding: quoted-printable
Date: Fri, 08 Jul 2016 16:45:24 +0200
To: mail.to@gmail.com
From: mail.sender@gmail.com
Subject: =?utf-8?B?44OG44K544OI44Oh44O844Or77yIdjE3IDQzNzIpIA==?=
```

```
=E6=97=A5=E6=9C=AC=E8=AA=9E=E3=81=A7=E7=B5=82=E3=82=8F=E3=82=8B
```

2. Enregistrez le fichier avec l'extension ".eml", par exemple "test.eml".
3. Double-cliquez sur le fichier et vous visualiserez l'email dans MS Outlook comme si vous le receviez de votre serveur de mail.

A propos du timestamp (horodatage)

Pour se conformer aux spécificités des tests unitaires, lorsque vous utilisez la commande **SMTP_QuickSend** avec un fichier de sortie, l'en-tête Date se présente toujours sous cette forme :

```
Date: Fri, 08 Jul 2016 16:45:24 +0200
```

Ainsi, les comparaisons de date n'échoueront pas dans les tests unitaires.

Note : Lorsque vous utilisez un hôte réel (comme smtp.gmail.com), l'en-tête Date est remplacé par un véritable timestamp.

Si vous souhaitez obtenir un véritable timestamp dans votre fichier de test, vous pouvez utiliser la commande **SMTP_Send**. Dans ce cas, vous pouvez appeler la commande **SMTP_Date** et ainsi fournir un en-tête Date réel avec **SMTP_Send**.