

libMAIL - Émetteur SMTP minimal.

Bibliothèque MS-Access

Version 1

Manuel d'utilisation, version 1.06

Copyright 2009-2013 Denis SCHEIDT



Cette création est mise à disposition selon le Contrat Paternité-Pas d'Utilisation Commerciale-Partage des Conditions Initiales à l'Identique 2.0 France disponible en ligne <http://creativecommons.org/licenses/by-nc-sa/2.0/fr/> ou par courrier postal à Creative Commons, 171 Second Street, Suite 300, San Francisco, California 94105, USA.



Les illustrations qui ne sont pas spécifiques à libMAIL proviennent de <http://openclipart.org>, bibliothèque d'images sous licence [CC0 Public Domain Dedication](http://creativecommons.org/licenses/by-nc-sa/2.0/fr/).

Table des matières

1 Introduction.....	5
1.1 Par où commencer ?.....	5
1.2 Environnement de création de la bibliothèque.....	5
2 Objectifs et limites.....	7
2.1 Commandes et options SMTP mises en œuvre.....	7
3 Licence.....	9
4 Nouveautés de la version 1.41.....	11
5 Installation.....	13
5.1 Configuration requise.....	13
5.2 Création de la bibliothèque.....	13
6 Manuel de référence.....	15
6.1 Fonctionnement du serveur.....	15
6.1.1 La table BoiteMail.....	15
6.1.1.1 Création de la table.....	15
6.1.1.2 Description des champs.....	16
6.1.2 Les commandes de base.....	17
6.2 Interface de programmation.....	17
6.2.1 Créer un courrier électronique.....	17
6.2.1.1 Corps Texte et HTML.....	21
6.2.1.2 Joindre un objet Access.....	23
6.2.1.3 Identifiant de message.....	24
6.2.1.4 Erreurs de pièces jointes.....	24
6.2.1.5 Taille des messages.....	25
6.2.2 Modifier un message existant.....	26
6.2.3 Démarrer le serveur.....	26
6.2.4 Options SMTP étendues.....	28
6.2.4.1 Options de message.....	28
6.2.4.1.1 Avis de remise (DSN).....	28
6.2.4.1.2 Accusés de réception et de lecture (MDN).....	29
6.2.4.1.3 Champs Origine.....	30
6.2.4.1.4 Priorité du message.....	30
6.2.4.2 Options de Serveur.....	30
6.2.5 Obtenir l'état du serveur.....	31
6.2.5.1 La variable d'état.....	32
6.2.5.2 Contenu de la variable.....	34
6.2.6 Contrôler le serveur.....	35
6.2.7 Limiter le menu.....	36
6.2.8 Gérer le journal.....	37
6.2.9 Commandes annexes.....	37
6.3 Interface utilisateur graphique.....	41
6.3.1 La zone de notifications.....	41
6.3.2 Contrôler le serveur.....	42
6.3.3 Édition interactive.....	43
6.3.4 Gérer la table BoiteMail.....	46
6.3.5 Visualiser l'état du serveur.....	47
6.3.6 Gérer le journal.....	48
7 Dépannage.....	49
7.1 Contrôler la connexion.....	49
8 Problèmes potentiels.....	51
8.1 Tâche de fond.....	51
8.2 Figement de l'interface.....	51
9 Contact.....	53
10 Historique des modifications.....	55
11 Références.....	57

1 Introduction

Il arrive régulièrement que l'on ait besoin d'envoyer un état ou un fichier (résultat d'une requête, par exemple) par messagerie électronique à partir d'une application Access.

Access n'offre que deux méthodes pour effectuer cette action : la méthode **SendObject** de l'objet **DoCmd** ou l'utilisation d'un objet **Outlook.Application** (si Outlook est le client de messagerie utilisé par le poste).

Ces deux solutions présentent l'avantage de la simplicité, mais ont également quelques inconvénients non négligeables :

- La méthode **SendObject** ne permet d'envoyer que des objets Access (formulaire, état, module, etc...). Il n'est pas possible de joindre un fichier externe. De plus, la mise en page est perdue, seules les données étant transmises.
- Lorsqu'une application est diffusée chez plusieurs clients, par exemple, rien ne permet d'être sûr qu'un poste disposera de MS-Outlook. Il peut utiliser un autre client de messagerie (qui ne sera peut-être pas compatible MAPI), ou même ne disposer d'aucun logiciel de messagerie électronique. Écrire un code tenant compte de tous les cas possibles peut vite devenir très compliqué.
- Piloter une application tiers pose également des problèmes : il faut s'assurer que l'application existe sur le poste, éventuellement l'inclure dans le jeu d'installation de l'application Access. Cette application risque d'être mise à jour par ailleurs, et perturber ainsi le déroulement de l'application Access.

Ces différentes contraintes peuvent nous faire préférer une solution plus souple et efficace : avoir un mini-serveur SMTP, écrit en VBA, donc utilisable directement depuis une application Access, et ne dépendant d'aucun composant ou logiciel externe.

1.1 Par où commencer ?

C'est la première fois que vous utilisez libMAIL, et vous êtes pressé de commencer...

Rassurez-vous, il n'est pas nécessaire de lire toute la documentation pour envoyer votre premier message, mais pour vous faire une idée du fonctionnement et des possibilités de libMAIL, vous devriez au moins lire les chapitres suivants :

Le chapitre 5 Installation, page 13, pour créer la version binaire de la bibliothèque, sans laquelle aucun envoi n'est possible.

Le chapitre 6.1 Fonctionnement du serveur, page 15, afin de prendre connaissance du fonctionnement de libMAIL.

Après toute cette partie théorique, il est temps d'envoyer un premier message. Vous pouvez utiliser le formulaire d'édition de message de libMAIL, présenté au chapitre 6.3.3 Édition interactive, page 43, pour le créer et l'envoyer sans avoir à écrire de ligne de code.

Lorsque vous aurez assimilé ces informations de base, vous pourrez ensuite vous attaquer aux autres chapitres de ce manuel, afin de profiter de toutes les fonctionnalités de libMAIL.

1.2 Environnement de création de la bibliothèque

Ce composant a été développé et testé sous MS-Access version 97. Pourquoi Access 97 ? Tout simplement parce que c'est la seule licence dont je dispose... L'avantage, c'est qu'il y a de fortes chances pour que le code fonctionne avec les versions suivantes.

Comme Access 97 ne doit plus être tellement répandu à l'heure actuelle (2008), la bibliothèque n'est pas fournie sous sa forme binaire (.mdb ou .mda), mais seulement sous forme de fichiers texte (code source) pouvant être chargés dans une base Access, afin de pouvoir compiler un complément adapté à la version que vous utilisez.

2 Objectifs et limites

libMAIL est un émetteur SMTP simplifié, développé sous forme de bibliothèque pour MS-Access, destiné à s'affranchir des contraintes et problèmes évoqués précédemment. Il implémente, de manière plus ou moins complète, les **RFC 1321, 2045, 2047, 2104, 2195, 2554, 2821, 2831, 3798 et 4616** [1] et [8]. Son seul but est de permettre l'envoi de messages électroniques, avec ou sans pièce(s) jointe(s), à partir d'une application. De ce fait, il ne comprend que le sous-ensemble de commandes nécessaire à cette tâche (voir tableau ci-dessous). Le cas échéant, la pièce jointe est encodée dans un message au format MIME [4]. Cette implémentation est elle aussi réduite au strict nécessaire pour permettre l'adjonction d'un fichier au message.

La bibliothèque peut être pilotée directement par une application, au travers de son interface de programmation (voir page 17), mais elle dispose également d'une interface graphique (voir page 41), matérialisée par une icône dans la zone de notifications, permettant une utilisation interactive.

2.1 Commandes et options SMTP mises en œuvre

Lors de sa communication avec le serveur SMTP distant, libMAIL reconnaît et emploie le sous-ensemble de commandes et options listées dans le tableau ci-dessous.

Quelques options étendues sont prises en charge, parmi les plus courantes. Si vous utilisez l'une ou l'autre de ces options, libMAIL s'assurera que le serveur distant les accepte. Dans le cas contraire, il tentera tout de même d'envoyer les messages en ignorant les options non supportées.

EHLO/HELO	Commande d'ouverture de session. Si EHLO est refusé par le serveur distant, libMAIL tente de se connecter avec la commande HELO. Dans ce cas, les extensions SMTP ne sont pas gérées.
AUTH	Authentification de la connexion, si les extensions SMTP sont activées. libMAIL prend en charge les méthodes LOGIN, PLAIN, CRAM-MD5 et DIGEST-MD5 .
MAIL FROM RCPT TO DATA	Ces commandes permettent d'envoyer les différents éléments constituant le message.
<i>Option SIZE</i>	Taille de message, si les extensions SMTP sont activées. Si le serveur distant prend cette extension en charge, libMAIL enverra une estimation de la taille du message dans la commande MAIL. Si le serveur distant doit rejeter le message parce qu'il dépasse la taille maximale autorisée, libMAIL en sera averti avant d'avoir commencé la transmission. Il invalidera le message et ne perdra plus de temps pas la suite en essayant de le transmettre à nouveau.
<i>Option DSN</i>	Gestion des avis de remise de messages, si les extensions SMTP sont activées.
RSET	Commande de réinitialisation. Le serveur distant annule la session en cours.
QUIT	Fermeture de la communication avec le serveur distant.

3 Licence

libMAIL est distribuée sous licence LGPL, version 3 ou supérieure.



Vous avez toute latitude pour la modifier et l'améliorer, vous pouvez l'utiliser dans vos applications, qu'elles soient sous licence libre ou propriétaire. Vous êtes par contre tenu de fournir les sources de la bibliothèque, incluant vos éventuelles modifications, avec votre application.

Le texte complet de la licence peut-être consulté ici : <http://www.gnu.org/licenses/lgpl.html>. Une copie de la licence est incluse dans les fichiers COPYING et COPYING.LESSER, que vous devez distribuer également.

Il n'existe pas de traduction officielle [6] de la licence LGPL en français. Seule la version anglaise de la licence est applicable.

libMAIL est fournie en l'état, sans garantie d'aucune sorte.

Important

La licence LGPL permet la liaison **dynamique** avec une application, quelle que soit la licence de cette application. Vous **devez** donc créer une bibliothèque si vous utilisez libMAIL avec une application soumise à une licence propriétaire.

Incorporer les formulaires et modules de libMAIL directement dans votre application est assimilé à une liaison **statique**, et a pour conséquence d'étendre la licence LGPL à toute l'application.

Vous pouvez par contre envisager d'incorporer le code de libMAIL dans votre application si celle-ci est sous licence libre.[5]

4 Nouveautés de la version 1.41

Cette version n'apporte pas de nouvelle fonctionnalité, mais corrige un certain nombre de bugs. Consultez le fichier `ChangeLog.txt` pour la liste des problèmes corrigés.

- Ajout des options Répondre à et Envoyeur aux fonctions de création de message. (voir page 30)
- Contrôle de l'IDEnvelope. (voir page 28)
- Adaptation des appels API pour les plates-formes Windows 64 bits.
- Commandes et fonctions modifiées : aucune.
- Nouvelles commandes et fonctions : Joindre, SMTPTest.

5 Installation

libMAIL est une simple bibliothèque. Aucune installation particulière n'est requise. Vous devez simplement lier la bibliothèque à votre application, à l'aide de la boîte de dialogue Références, si vous désirez l'utiliser. Mais avant, il faut créer le fichier binaire à partir des sources.

5.1 Configuration requise

- Access 97 ou supérieur
- Windows 2000 ou supérieur

Les références suivantes :

- DAO (Microsoft DAO Object Library)
- MSOffice X.y Object Library (msoxx.dll) (pour la manipulation des CommandBars)

5.2 Création de la bibliothèque

- Créez un dossier sur votre disque dur ;
- Copiez les fichiers .FRM et .MOD dans ce dossier ;
- Créez une base de données vierge dans ce dossier. Renommez-la en libMAIL.MDB ;
- Ouvrez cette base ;
- Créez un nouveau module et collez ou importez-y le contenu du fichier Outils.MOD.
Attention aux lignes **Option Compare Database** et **Option Explicit** qui sont insérées automatiquement en début de module. Supprimez-les pour éviter une erreur de compilation.
Enregistrez le module sous le nom Outils. Ce module contient une procédure qui **efface tous les modules et formulaires de la base**, sauf le module nommé Outils. **Il est donc important d'enregistrer ce premier module sous ce nom précis**, afin d'éviter que la procédure ne se supprime elle-même au cours de son exécution, provoquant l'arrêt brutal d'Access.
La procédure importe tous les fichiers *.FRM et *.MOD présents dans le répertoire contenant la base de données courante.
- Ouvrez la fenêtre de débogage (ou fenêtre d'exécution) (Ctrl-G) ;
- Dans cette fenêtre, tapez simplement la commande : ChargeVB
Cette commande va tenter de définir les références nécessaires, charger tous les fichiers sources, établir le nom de projet et le compiler automatiquement.
Si vous obtenez une erreur de compilation à ce stade, activez l'option de compilation à la demande, puis relancez la commande.
Si la commande ne peut toujours pas être lancée, établissez les références nécessaires à l'aide la boîte de dialogue Références, puis relancez-la ;

Vous pouvez maintenant placer le fichier obtenu où bon vous semble.

Remarque

*libMAIL utilise la bibliothèque DAO (Microsoft DAO Object Library). Cette bibliothèque est référencée par défaut dans Access 97, mais pas obligatoirement dans les versions plus récentes.
La manipulation des objets CommandBars demande une référence à la bibliothèque MSOffice X.y Object Library (X.y représentant la version d'Access). Sous Access 97, ce fichier s'appelle mso97.dll. Dans les versions plus récentes, il est nommé mso.dll.
Si vous obtenez une erreur de compilation, pensez à vérifier ces points.*



Retrouvez toute la procédure de création de la bibliothèque en vidéo sur Self-Access.com : http://grenier.self-access.com/public/videos/ac_libmail.swf

Vous pouvez bien entendu utiliser directement la base créée à l'étape précédente comme bibliothèque pour vos applications. Toutefois, si vous voulez distribuer votre application sous forme de fichiers MDE, il faudra enregistrer la bibliothèque dans ce format également, sinon Access refusera de 'compiler' votre application.

Une bonne pratique consiste donc à utiliser une bibliothèque uniquement sous forme 'compilée' (fichier MDE).

6 Manuel de référence

6.1 Fonctionnement du serveur

Le serveur SMTP de libMAIL ne diffère pas tellement d'un client de messagerie classique dans son fonctionnement, si l'on excepte le fait qu'il ne peut qu'émettre des courriels et non en recevoir. Dans un premier temps, il faut créer le ou les messages, puis déclencher l'envoi à l'aide la commande appropriée. Dans votre client de messagerie, vous cliqueriez d'abord sur 'Nouveau message', puis, une fois que le message est enregistré, sur le bouton 'Envoyer/Recevoir'.

libMAIL se sert d'une table (**BoiteMail**) pour stocker les messages en attendant de les envoyer. On pourrait écrire directement dans cette table pour y créer des messages. Si cette solution est envisageable (bien que non recommandée) pour un message constitué simplement d'un corps en texte brut, l'opération est déjà nettement plus compliquée lorsqu'il faut joindre un ou plusieurs fichiers au courriel, car dans ce cas le message doit respecter une structure particulière pour être compris des clients de messagerie des destinataires.

Les fonctions de création de message feront ce travail pour vous. Vous trouverez leur description complète au chapitre 6.2.1, page 17.

Ces fonctions ne font qu'écrire un enregistrement dans la table BoiteMail, mais elles n'envoient pas les messages. Pour déclencher leur expédition, un second jeu de fonctions permet de démarrer le serveur SMTP, en lui fournissant les informations nécessaires à la connexion.

Contrairement au client de messagerie classique qui stocke les informations de connexion dans des comptes de messagerie, libMAIL ne les mémorise pas. Vous devez les fournir lors de chaque utilisation de la commande de démarrage du serveur. Si ces informations doivent évoluer au fil du temps, il peut être utile de prévoir une table dans votre application, ou de les stocker dans la base de registre ou dans un fichier texte, par exemple, plutôt que de les saisir 'en dur' dans votre code.

Les commandes de démarrage provoquent le chargement du formulaire frm_SMTP, dont la seule partie visible est l'icône qui s'affiche dans la zone de notification de Windows. C'est ce formulaire qui enverra les messages vers le serveur du fournisseur d'accès. Ces fonctions sont décrites au chapitre 6.2.3, page 26.

Ce formulaire va chercher dans la table les messages dont l'état est 'E', les envoyer, changer l'état en 'V', puis se refermer.

Remarque

Il est possible d'empêcher la fermeture du formulaire après l'envoi à l'aide d'un paramètre de la commande de lancement du serveur. Celui-ci scrutera alors la table à intervalles réguliers.

6.1.1 La table BoiteMail

6.1.1.1 Création de la table

La table BoiteMail ne fait pas partie de la bibliothèque. Elle doit être créée dans votre application, soit directement dans la base contenant le code qui va utiliser l'émetteur SMTP, soit en tant que table attachée dans cette même base. Le code de la bibliothèque fait référence à cette table à l'aide de CurrentDb.

Lorsque vous tentez de créer un message (à l'aide de l'une des fonctions **CreeMail**), ou lorsque vous démarrez le serveur (à l'aide de la fonction **(E)SMTPLance**), libMAIL vérifie l'existence de la table BoiteMail, et vous propose de la créer le cas échéant.

Si vous le désirez, vous pouvez appeler la fonction **VerifieBAL** vous-même afin de créer la table. Aucun paramètre n'est requis. La fonction retourne True si la création réussit, False si elle échoue.

La fonction **VerifieBAL** ne **créera pas** la table dans la bibliothèque elle-même. Elle doit donc être exécutée depuis votre application, soit par un appel explicite, soit implicitement lors d'un premier appel à **CreeMail** ou **SMTPLance**.

Si vous désirez changer le nom de la table, vous pouvez éditer le module **prvGlobales** avant d'appeler **VerifieBAL** pour la première fois. Modifiez la constante BoiteMail.

6.1.1.2 Description des champs

Vous trouverez ci-dessous sa structure. En fonction des besoins, vous pouvez ajouter d'autres champs, mais ceux décrits ci-dessous doivent impérativement figurer dans la table, car ils sont utilisés par la bibliothèque.

Table: BoiteMail

Columns

Name	Type	Size
-----	-----	-----
Identifiant	Texte	18
Utilisateur	Texte	25
DateMsg	Date/Heure	8
Etat	Texte	1
Expediteur	Texte	255
Destinataires	Mémo	-
CC	Mémo	-
BCC	Mémo	-
Objet	Texte	255
CorpsMsg	Mémo	-
ESMTP	Mémo	-
Differer	Date/Heure	8
Conserver	Date/Heure	8
DateEnvoi	Date/Heure	8

Champ	Description	Notes
Identifiant	Identifiant unique du message, créé automatiquement par la fonction CreeMail . Cet identifiant est aussi la clé primaire de la table.	1
Utilisateur	Nom de l'utilisateur qui a créé le message.	
DateMsg	Date de création du message. Ce champ est renseigné automatiquement par libMAIL lors de la création du message à l'aide des commandes CreeMail.	
Etat	E = message en boîte d'envoi (à envoyer). V = message envoyé. libMAIL écrit cette valeur après un envoi réussi. X = message rejeté définitivement par le serveur distant (taille supérieure au maximum autorisé, par exemple). Dans ce cas, libMAIL ne tentera plus d'envoyer le message. D = message marqué comme supprimé. M = message en cours de modification par l'utilisateur (voir ModifieMail , page 26).	
Expediteur	Adresse mail de l'expéditeur. Cette adresse doit être connue du serveur SMTP auquel libMAIL se connecte.	
Destinataires	Liste des destinataires du message, séparés par des virgules ou des point-virgules.	
CC	Liste des destinataires pour copie.	
BCC	Liste des destinataires pour copie invisible.	
Objet	Texte inséré dans l'objet du message électronique.	
CorpsMsg	Corps complet du message, avec les éventuelles pièces jointes au format MIME. Si le message ne comporte pas de pièce jointe, ce champ contient simplement le texte inséré dans le corps du mail. Vous trouverez un exemple de corps de message à la fin du chapitre 6.2.1	
ESMTP	Options SMTP étendues liées aux messages. Ce champ contient une chaîne de caractères codée en Base64, représentant les options à ajouter au message. Les options d'accusé de réception (DSN) sont transmises par l'intermédiaire de ce	1

Champ	Description	Notes
	champ.	
Differer	Envoi différé du message. Le message sera envoyé lors de la première scrutation intervenant après la date/heure spécifiée dans ce champ.	2
Conserver	Ne pas effacer le message de la table avant cette date/heure. Ce champ est utilisé par la fonction Purge lors de l'effacement des messages en état 'V'.	2
DateEnvoi	Date et heure du dernier envoi du message. Contient 0 si le message n'a pas été envoyé ou s'il a été modifié.	2

Notes :

- 1 : Utilisé à partir de la version 1.20 de libMAIL (version 2 de la table).
- 2 : Utilisé à partir de la version 1.40 de libMAIL (version 3 de la table).

Remarque

*Les messages ne sont pas supprimés automatiquement de la table. Ceci implique qu'il faudra la nettoyer régulièrement pour réduire son occupation disque. Vous pouvez utiliser la fonction **Purge** de la bibliothèque (voir page 40).*

6.1.2 Les commandes de base

Pour envoyer un message depuis une application, vous devrez faire appel à deux fonctions de libMAIL :

EcreeMailMIME (ou l'une de ses déclinaisons) pour créer les messages dans la table BoiteMail.

ESMTP Lance (ou l'une de ses déclinaisons) pour démarrer le serveur et provoquer l'envoi proprement dit.

Ces deux commandes sont suffisantes pour effectuer des envois simples de courriels.

Les fonctions de création des messages sont décrites au chapitre 6.2.1 Créer un courrier électronique, page 17. Lorsque vous aurez créé vos messages, passez au chapitre 6.2.3 Démarrer le serveur, page 26 pour apprendre comment les envoyer.

Vous aurez aussi besoin :

- du nom (ou de l'adresse IP) du serveur SMTP auquel libMAIL va se connecter. En règle générale, c'est le serveur SMTP de votre Fournisseur d'Accès Internet ou le serveur de messagerie interne de votre entreprise ;
- de votre login et mot de passe si ce serveur nécessite une authentification. Vous devrez alors utiliser **ESMTP Lance** (car **SMTP Lance** ne gère pas l'authentification).

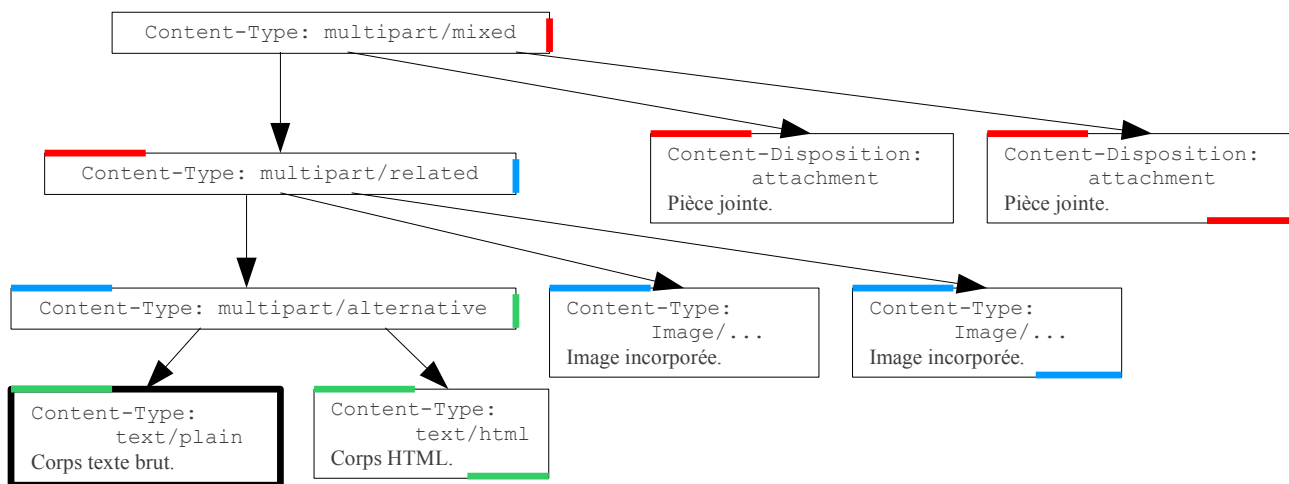
6.2 Interface de programmation

Ce chapitre détaille les procédures et fonctions que vous pouvez appeler depuis votre application afin de créer des messages et contrôler le serveur.

6.2.1 Créer un courrier électronique

Un message au format MIME se présente sous la forme d'un arbre pouvant comporter un nombre indéterminé de niveaux.

Le schéma ci-dessous représente la structure complète que libMAIL peut générer pour un message comprenant à la fois un corps texte brut, un corps HTML comportant des images, et des fichiers joints.



La partie text/plain représentant le corps en texte brut figurera toujours dans un message, même vide. Les autres parties ne seront ajoutées que si c'est nécessaire.

Les fonctions de création de message se déclinent en plusieurs versions, permettant de créer, au choix, un message simple, avec des options étendues ou avec une partie formatée en HTML.

Si vous ne savez pas quelle fonction employer, choisissez **ECreeMailMIME**. C'est la plus complète et elle vous permettra de créer tous les messages, avec toutes les options possibles.

ECreeMail diffère de la précédente en ce qu'elle ne permet pas de spécifier un corps de message enrichi (HTML). Seul le corps en texte brut est disponible. Les options étendues sont permises.

CreeMail est la version la plus simple, puisqu'elle n'accepte ni corps HTML, ni option étendue.

Toutes ces fonctions permettent de joindre des fichiers aux messages.

Chaque fonction renseigne les champs de la table BoiteMail avec les éléments passés en paramètres et crée le corps de message, incluant la/les pièce(s) jointe(s) et le corps HTML le cas échéant. Elle retourne un identifiant unique pour le message qui vient d'être créé (voir page 24).

Ces fonctions ne font que créer le message dans la table, elles ne démarrent pas automatiquement le serveur.

Remarque

CreeMail et ECreeMail font appel à ECreeMailMIME avec des valeurs par défaut pour OptionsESMTP et la partie MIME du message.

Syntaxe

ECreeMailMIME(Destinataires, ObjetMsg, dtuTexteMessage, Expéditeur, OptionsESMTP, [Utilisateur], [CC], [BCC], [PiècesJointes()], [EditeMail], [Differer], [Conserver])

ECreeMail(Destinataires, ObjetMsg, TexteMessage, Expéditeur, OptionsESMTP, [Utilisateur], [CC], [BCC], [PiècesJointes()], [EditeMail], [Differer], [Conserver])

CreeMail(Destinataires, ObjetMsg, TexteMessage, Expéditeur, [Utilisateur], [CC], [BCC], [PiècesJointes()], [EditeMail], [Differer], [Conserver])

Élément	Description
<i>Destinataires</i>	Chaîne. Liste des destinataires, séparés par des virgules ou des point-virgules. Doit contenir au moins une adresse de destinataire.
<i>ObjetMsg</i>	Chaîne. Texte qui figurera dans le champ Objet du message.

<i>TexteMessage</i>	Chaîne. Le texte à l'intention du destinataire du message.
<i>dtuTexteMessage</i>	<p>Dans le cas de la fonction ECreeMailMIME, ce paramètre est une variable du type utilisateur tuMessageMIME, comprenant deux membres :</p> <p>Texte : contient le corps de message au format texte brut ;</p> <p>HTML : contient la version HTML du corps du message.</p> <p>Ces deux membres sont de type chaîne de caractères. Texte contient la version texte brut du message, alors que HTML accueille la version HTML. Reportez-vous au chapitre 6.2.1.1 page 21 pour plus de précisions.</p>
<i>Expéditeur</i>	Chaîne. Adresse électronique de l'expéditeur. La plupart des serveurs SMTP vérifient cette adresse et rejettent le message si elle leur est inconnue. Utilisez une adresse connue de votre FAI.
<i>OptionsESMTP</i>	Cette option n'est requise que pour les commandes ECreeMail et ECreeMailMIME . Elle permet de passer au serveur des options étendues. Voir page 28 pour une description des options étendues propres aux messages.
<i>[Utilisateur = LoginWindows]</i>	Chaîne. Nom de l'utilisateur ou du poste de travail ayant créé le message. Cette information n'est utilisée que dans le journal. Ce paramètre est facultatif. Par défaut, c'est le login Windows qui sera utilisé.
<i>[CC = ""]</i>	Chaîne. Liste des destinataires en copie, séparés par des virgules ou des point-virgules. Ce paramètre est facultatif.
<i>[BCC = ""]</i>	Chaîne. Liste des destinataires en copie cachée, séparés par des virgules ou des point-virgules. Ce paramètre est facultatif.
<i>[PiecesJointes()]</i>	<p>Tableau de type Chaîne.</p> <p>Ce tableau comporte deux colonnes.</p> <p>La première contient seulement le nom de la pièce jointe, tel qu'il apparaîtra dans le message. Ce nom peut éventuellement être différent du nom réel du fichier.</p> <p>La seconde colonne contient le chemin d'accès complet à la pièce jointe s'il s'agit d'un fichier, ou une instruction spéciale s'il s'agit d'un objet Access (voir page 23).</p> <p>Le tableau doit être déclaré sous la forme Dim LeTableau(1, Lignes). Les lignes sont en dernière dimension, afin de permettre le redimensionnement du tableau avec ReDim Preserve en cas de besoin. Voir également la fonction PJFichier.</p> <p>Si l'élément dans la colonne 0 du tableau est une chaîne vide, la ligne sera ignorée (le fichier ne sera pas joint).</p> <p>Ce paramètre est facultatif.</p>
<i>[EditeMail = False]</i>	<p>Booléen. Si True, affiche un formulaire permettant de saisir ou modifier un message. Ce paramètre est facultatif et vaut False par défaut.</p> <p>Le message peut être enregistré dans la table et envoyé immédiatement à partir du formulaire.</p> <p>Voir page 43 pour une description du formulaire de saisie de message.</p>
<i>[Differer = 0]</i>	<p>Date/Heure. Le message ne sera pas envoyé avant la date/heure spécifiée.</p> <p>Ce paramètre est facultatif. Par défaut, il vaut 0 (l'envoi du message n'est pas différé).</p>
<i>[Conserver = 0]</i>	<p>Date/Heure. Lors de l'appel à la fonction Purge, le message ne sera effacé qu'après la date/heure spécifiée, et seulement s'il a été envoyé (état 'V').</p> <p>Ce paramètre est facultatif. Par défaut, il vaut 0 (Purge effacera le message).</p>

Exemple

Cet exemple utilise des options étendues décrites en page 28.

```

' Initialiser la liste des pièces jointes
Dim tblPJ(1, 1) As String

' Variable pour les options étendues de message.
Dim dtuOptions As tuESMTP_MSG

' Variable récupérant l'identifiant du message créé.
Dim sIDMsg As String

' Tableau bidimensionnel des pièces jointes.
' Le premier indice désigne la colonne, le second la ligne (une ligne par P.J.)
' Colonne 0 : nom de la PJ tel qu'il apparaîtra dans le message.
' Colonne 1 : chemin d'accès complet au fichier à joindre.
tblPJ(0, 0) = "PJ1.txt" : tblPJ(1, 0) = "C:\Temp\PJ1.TXT"
tblPJ(0, 1) = "PJ2.txt" : tblPJ(1, 1) = "C:\Temp\PJ2.TXT"

' Options d'accusés de réception.
With dtuOptions.DSN
    ' Le serveur SMTP distant doit envoyer une notification en cas d'échec
    ' ou de remise différée...
    .Notification = lmESMTPDsnNotifEchec + lmESMTPDsnNotifDelai
    ' ... et ne retourner que l'en-tête du message d'origine.
    .Retour = lmESMTPDsnRetHdrs
    .IDEnvelope = "IDMessage000001"
End With

' Créer le mail dans la table BoiteMail
sIDMsg = ECreeMail("adresse.mail@destinataire.fr", _
    "Réservations du " & Date, _
    "Veuillez trouver ci-joint l'état des réservations...", _
    "adresse.mail@expediteur.fr", _
    dtuOptions, _
    CurrentUser, _
    , _
    , _
    tblPJ())

...

' Structure accueillant les différentes versions du corps du message.
Dim dtuMsgMIME As tuMessageMIME

With dtuMsgMIME
    .Texte = "Veuillez trouver ci-joint l'état des réservations..." & vbCrLf & _
        "Cordialement, ..."
    .HTML = "<HTML>" & vbCrLf & _
        "<P><B>Veuillez trouver ci-joint l'état des réservations...</B></P>" &
        vbCrLf & _
        "<P>Cordialement, ...</P>" & vbCrLf & _
        "</HTML>"
End With

' Créer le mail dans la table BoiteMail
sIDMsg = ECreeMailMIME("adresse.mail@destinataire.fr", _
    "Réservations du " & Date, _
    dtuMsgMIME, _
    "adresse.mail@expediteur.fr", _
    dtuOptions, _
    CurrentUser, _
    , _
    , _
    tblPJ())

```

Le message existe maintenant sous forme d'enregistrement dans la table, mais il ne sera pas envoyé tant que le serveur n'aura pas été lancé.

Voici le contenu du champ CorpsMsg de la table, créé par l'appel à **ECreeMailMIME** de l'exemple ci-dessus :

```

MIME-Version: 1.0 (=utf-8?Q?Biblioth=C3=A8que_VBA_libMAIL_v.1.40?=)
Content-Type: multipart/mixed; boundary="Separateur_3518726400.122448"

Ce message au format MIME comporte plusieurs parties.

```


employée dans l'exemple précédent.

- Dans le cas contraire, utilisez un éditeur HTML pour créer une page qui pourra ensuite être chargée dans le membre HTML.
Vous pouvez utiliser votre traitement de texte habituel pour cette tâche, dans la mesure où la plupart d'entre eux savent enregistrer ou exporter au format HTML.
Lorsque votre message est prêt, sauvegardez-le sur votre disque dur. Vous pouvez alors initialiser le membre HTML à l'aide du **chemin d'accès complet** à ce fichier (sous la forme **x:\Chemin\Fichier.html**, ou **\\Serveur\Partage\Chemin\Fichier.html**). libMAIL lira le contenu de ce fichier et s'en servira pour créer la partie HTML du message.

Remarque

Dans les deux cas, si vous laissez le membre **Texte** vide (""), libMAIL convertira le membre **HTML** en texte brut et complètera automatiquement le membre **Texte**.

Exemple :

```
' Structure accueillant les différentes versions du corps du message.
Dim dtuMsgMIME As tuMessageMIME

dtuMsgMIME.HTML = "C:\Temp\MessageReservations.html"

' Créer le mail dans la table BoiteMail
sIDMsg = ECreeMailMIME("adresse.mail@destinataire.fr", _
    "Réservations du " & Date, _
    dtuMsgMIME, _
    "adresse.mail@expediteur.fr", _
    dtuOptions, _
    CurrentUser, _
    , _
    , _
    tbLPJ())
```

Dans cet exemple, MessageReservations.html contient le code HTML suivant, généré ici par LibreOffice :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <META HTTP-EQUIV="CONTENT-TYPE" CONTENT="text/html; charset=utf-8">
  <TITLE></TITLE>
  <META NAME="GENERATOR" CONTENT="LibreOffice 3.3 (Linux)">
  <META NAME="AUTHOR" CONTENT="Denis ">
  <META NAME="CREATED" CONTENT="20110720;15134400">
  <META NAME="CHANGEDBY" CONTENT="Denis ">
  <META NAME="CHANGED" CONTENT="20110720;15154700">
  <STYLE TYPE="text/css">
  <!--
    @page { margin: 2cm }
    P { margin-bottom: 0.21cm }
  -->
  </STYLE>
</HEAD>
<BODY LANG="fr-FR" DIR="LTR">
<P STYLE="margin-bottom: 0cm">Veuillez trouver ci-joint l'état des
réservations ...</P>
<P STYLE="margin-bottom: 0cm"><BR>
</P>
<P STYLE="margin-bottom: 0cm">Cordialement, ...</P>
</BODY>
</HTML>
```

Remarques

- Seuls les 64000 premiers caractères du fichier HTML seront lus par libMAIL.
- Les traitements de textes ajoutent souvent une « pollution » plus ou moins importante, constituée d'informations totalement inutiles dans le cas d'un courrier électronique, généralement contenue dans une balise <HEAD>. libMAIL supprimera automatiquement cette balise.
- libMAIL n'ouvre que des fichiers portant l'extension .html ou .htm.
- Le document HTML peut contenir des images.

6.2.1.2 Joindre un objet Access

La colonne d'indice 1 du tableau des pièces jointes accepte une instruction spéciale en lieu et place de la spécification de fichier, permettant de joindre directement un objet Access (Table, Formulaire, État, etc...) au message.

Cette instruction est en réalité une simple chaîne de caractères qui va permettre de désigner l'objet à envoyer, ainsi que le format sous lequel il doit être exporté. Cette chaîne comporte 4 champs, séparés par '/' :

TE/TO/Format/Nom Objet

Champ	Valeur	Description
TE	<i>ImIPJDonnees</i> ou <i>ImIPJSource</i>	<i>ImIPJDonnees</i> : libMAIL utilise la commande DoCmd.OutputTo pour créer le fichier à joindre au message. <i>ImIPJSource</i> : libMAIL utilise la commande Application.SaveAsText pour créer le fichier. Dans ce cas, c'est la définition de l'objet (son code source) qui est jointe au message, et non les données sous-jacentes de l'objet.
TO	Toute constante valide pour la version d'Access considérée.	Détermine le type d'objet à joindre. Pour TE= <i>ImIPJDonnees</i> , ce sera l'une des constantes <i>acOutputForm</i> , <i>acOutputReport</i> , etc... Pour TE= <i>ImIPJSource</i> , les constantes sont <i>acForm</i> , <i>acReport</i> , etc...
Format	Toute constante valide pour la version d'Access considérée.	Détermine le format de la pièce jointe. Pour TE= <i>ImIPJDonnees</i> , ce sera l'une des constantes <i>acFormatXLS</i> , <i>acFormatRTF</i> , etc.. Ce champ est ignoré pour TE= <i>ImIPJSource</i> .
Nom objet	Chaîne de caractères	Nom de l'objet de base de données à envoyer.

Selon la version d'Access, la liste des objets pouvant être exportés ainsi que les formats acceptables pour chaque type d'objet changent. Reportez-vous à l'aide de la commande **DoCmd.OutputTo** pour connaître toutes les valeurs et combinaisons de constantes possibles.

La fonction **PJOA** peut créer automatiquement pour vous cette chaîne de caractères à partir des paramètres qui lui sont fournis.

Syntaxe :

PJOA(*TypeObjet*, *FormatExport*, *NomObjet*, [*TypeExport*])

Élément	Description
<i>TypeObjet</i>	Entier . Une constante intrinsèque d'Access, permettant de désigner le type d'objet : Formulaire, Table, Requête, etc... Ce sont des constantes de la forme <i>acOutputXXX</i> pour un export de type <i>ImIPJDonnees</i> , ou <i>acXXX</i> pour un export de type <i>ImIPJSource</i> . Où <i>XXX</i> représente le nom de la classe d'objets : Form, Report, Query, etc... Consultez l'aide d'Access pour connaître la liste des constantes autorisées
<i>FormatExport</i>	Chaîne : l'une des constantes <i>acFormatXXX</i> autorisées pour la commande DoCmd.OutputTo . Cet élément est ignoré pour un export de type <i>ImIPJSource</i> .

Consultez l'aide d'Access pour connaître la liste des constantes autorisées

NomObjet

Chaîne : Nom de l'objet tel qu'il apparaît dans la fenêtre de base de données.

*[TypeExport =
lmlPJDonnees]*

Entier. Type d'export.

lmlPJDonnees pour exporter les données de la table, du formulaire, de la requête. Cet export est fait à l'aide de la commande **DoCmd.OutputTo**.

lmlPJSource pour exporter la définition de l'objet à l'aide de **Application.SaveAsText**. Ce fichier source peut être importé dans une base de données pour y recréer l'objet.

Ce paramètre est optionnel et vaut *lmlPJDonnees* par défaut.

Exemple :

Le code ci-dessous prépare le tableau pour deux pièces jointes. La première est un fichier disque, la seconde comportera un fichier au format Excel, contenant les données affichées par le formulaire frm_Stock.

```
Dim PJ(1, 1) As String
```

```
PJ(0, 0) = "P.J. 1": PJ(1, 0) = "C:\Temp\MDBs.TXT"
```

```
PJ(0, 1) = "P.J. 2": PJ(1, 1) = PJOA(acForm, acFormatXLS, "frm_Stock", lmlPJDonnees)
```

PJOA() correspond à la chaîne suivante :

```
"0/2/Microsoft Excel (*.xls)/frm_Stock"
```

6.2.1.3 Identifiant de message

Les fonctions de création de messages retournent une chaîne de dix-huit caractères représentant l'identifiant du message créé. Cet identifiant est unique, il constitue la clé primaire de la table.

Les quatorze premiers caractères sont tirés de la date et de l'heure de création (année, mois, jour, heure, minute, seconde). Les quatre derniers sont aléatoires. Sa forme est la suivante :

AAAAMMJJHHNNSSxxxx

6.2.1.4 Erreurs de pièces jointes

Le tableau des pièces jointes peut être modifié par **(E)CreeMail** pour indiquer qu'un ou plusieurs fichiers n'ont pu être joints au message.

Lorsqu'un fichier n'a pu être joint au message, le contenu de la première colonne du tableau (d'indice zéro) est modifié par insertion du code d'erreur devant le nom de la pièce jointe.

L'exemple ci-dessous illustre la modification du tableau pour un fichier n'ayant pu être trouvé. Le code d'erreur (ici 53, Fichier non trouvé) est inséré devant le nom de la pièce jointe, formaté sur 5 positions et suivi d'un double-point.

	Col. 0 : Nom de PJ	Col. 1 : Chemin d'accès complet
Ligne du tableau avant l'appel	Fichier1.txt	C:\TEMP\Fichier1.txt
Ligne du tableau après l'appel	00053:Fichier1.txt	C:\TEMP\Fichier1.txt

Votre programme peut ainsi déterminer quels fichiers n'ont pas été joints au message, et pour quelle raison, en parcourant le tableau. Vous pouvez, par exemple, utiliser l'expression suivante pour déterminer si un élément commence par un code d'erreur :

```
tblPJ(0, i) Like "#####:*
```

Si cette expression vaut True, l'élément de tableau commence par un code d'erreur.

La fonction **ErreursPJ** analyse la première colonne du tableau des pièces jointes afin de déterminer si tous les fichiers ont été joints au message ou non.

Si sa valeur de retour est zéro, tous les fichiers ont été joints correctement au message. Une valeur supérieure à zéro indique le nombre de fichiers qui n'ont pas été joints.

Syntaxe

ErreursPJ(*PiecesJointes()*, [*RAZ*])

Élément	Description
<i>PiecesJointes()</i>	Tableau de type Chaîne. C'est le tableau des pièces jointes qui a été transmis à (E)CreeMail.
[<i>RAZ</i> = False]	Booléen. Si mis à True, la fonction rétablit la première colonne du tableau en effaçant le code d'erreur. Ce paramètre est facultatif et vaut False par défaut.

Lorsque la pièce jointe est un fichier disque, il est aisé de connaître la cause de l'erreur à l'aide de la fonction VBA **Error\$(N°Erreur)**.

Dans le cas d'un objet Access, le numéro d'erreur retourné par la commande **DoCmd.OutputTo** ou **Application.SaveAsText** se traduit systématiquement par le message "Erreur définie par l'application ou par l'objet".

Le tableau ci-dessous donne une description un peu plus explicite des différentes erreurs pouvant se produire.

Erreur	Description
OutputTo	
2282	Incompatibilité entre le type d'objet et le format demandé. Par ex., il n'est pas possible d'exporter un module dans un format autre que <i>acFormatTXT</i> .
2289	Le chemin et/ou le nom de fichier est invalide.
2487	Valeur invalide pour le type d'objet, ou le type d'objet n'est pas autorisé. Par ex., <i>acTable</i> n'est pas un type d'objet valide pour la commande SaveAsText .
2501	Aucun objet portant le nom demandé n'a été trouvé dans la base de données.
3011	L'objet existe dans la base de données, mais ne correspond pas au type spécifié.
SaveAsText	
2487	Voir ci-dessus.
2950	Le chemin et/ou le nom de fichier est invalide.
2956	Aucun objet portant le nom demandé n'a été trouvé dans la base de données.

6.2.1.5 Taille des messages

Si le serveur distant prend en charge l'extension **SIZE**, libMAIL ajoutera une estimation de la taille du message dans la commande **MAIL FROM**. Si la réponse du serveur indique qu'il ne dispose pas, temporairement, des ressources suffisantes pour traiter ce message (code de retour 4xx), libMAIL laissera le message dans l'état 'E'. Le message sera envoyé lors de la scrutation suivante.

Si le serveur indique que la taille du message est supérieure à ce qu'il peut accepter (code de retour 5xx), libMAIL passera ce message en état 'X' (invalidé), et ne tentera plus de l'envoyer.

6.2.2 Modifier un message existant

La commande **ModifieMail** permet la modification interactive d'un message existant.

Seuls les messages dans l'état 'E' (boîte d'envoi) peuvent faire l'objet d'une modification. La commande ne permet pas la modification d'un message se trouvant dans un état différent.

Cette commande n'a pas d'effet non plus lorsqu'un envoi est en cours (lorsque le serveur est dans l'état *Im/SrvEnCours*).

Avant d'appeler le formulaire d'édition de message, la commande change l'état du message (il passe dans l'état 'M'), afin qu'il ne soit pas sélectionné si le serveur démarre pendant la durée de l'édition.

À la fin de l'édition, le message est rétabli dans l'état 'E', et pourra être expédié lors de la scrutation suivante.

Syntaxe

ModifieMail(*Identifiant*)

La commande ouvre le formulaire d'édition de message, en chargeant le message demandé.

Si aucun message ne correspond à l'identifiant passé en paramètre, la commande échoue et affiche un message d'erreur.

6.2.3 Démarrer le serveur

La commande ci-dessous permet d'ouvrir le formulaire frm_SMTP. Ce formulaire va lancer une requête sur la table BoiteMail. S'il trouve au moins un enregistrement dans l'état 'E' (boîte d'envoi), il va établir une connexion avec le serveur distant, s'authentifier auprès de lui (si cela a été demandé), puis tenter d'expédier les messages. Si l'envoi réussit, le formulaire passera l'état des enregistrements à 'V'. Il se déconnectera ensuite du serveur distant puis, suivant la valeur du paramètre *EnvoieQuitte*, il se remettra en attente ou se fermera.

La commande admet deux formes différentes. Vous utiliserez **SMTPLance** lorsqu'aucune option étendue n'est requise. Si, au contraire, vous voulez utiliser des paramètres étendus (authentification), vous devrez utiliser la commande **ESMTPLance**.

La seule différence entre les deux commandes réside dans la présence d'un paramètre obligatoire supplémentaire pour **ESMTPLance**.

Remarques

- **SMTPLance** fait appel à **ESMTPLance** avec des valeurs par défaut pour *OptionsESMTP*.
- Si vous voulez passer de nouvelles options au formulaire, il n'est pas nécessaire de le télécharger avant de rappeler **(E)SMTPLance**.
- Le formulaire n'est pas visible.
- Cette commande effectue une RàZ de la variable d'état du serveur.

Syntaxe

ESMTPLance(*NomSrv*[:*Port*], *OptionsESMTP*, [*HELOdomain*], [*LogData*], [*LogComm*], [*FichJnl*], [*EnvoiQuitte*], [*DelaiVerif*], [*DelaiReponse*])

SMTPLance(*NomSrv*[:*Port*], [*HELOdomain*], [*LogData*], [*LogComm*], [*FichJnl*], [*EnvoiQuitte*], [*DelaiVerif*], [*DelaiReponse*])

- ! À partir de la version 1.30, les positions des paramètres *OptionsESMTP* et *HELOdomain* de la commande **ESMTPLance** sont interverties et *HELOdomain* est facultatif.

Élément

Description

NomSrv[:*Port* = 25]

Chaîne. Nom ou adresse IP (sous la forme a.b.c.d) du serveur SMTP auquel libMAIL devra se connecter pour envoyer les messages. En règle générale, c'est le nom du serveur SMTP de votre FAI.

Si votre serveur SMTP utilise un port différent du port standard (25), vous pouvez préciser cette valeur, précédée d'un double-point, à la suite du nom du serveur.

OptionsESMTP

Cette option n'est requise que pour la commande **ESMTP Lance**.

Elle permet de passer au serveur des options étendues. Voir page 30 pour une description des options étendues propres au serveur.

[HELODomain = NomDuPC] **Chaîne**. Le nom avec lequel libMAIL va s'identifier auprès du serveur SMTP désigné par *NomSrv*. Vous pouvez utiliser votre nom de domaine si vous en avez un, ou le nom de domaine de votre FAI.

Certains serveurs SMTP effectuent un contrôle sur ce paramètre, et peuvent rejeter la connexion s'il est jugé invalide.

Ce paramètre est facultatif. S'il n'est pas fourni, c'est le nom du PC qui est utilisé.

[LogData = False]

Booléen. Indique si les commandes DATA du protocole doivent être écrites dans le fichier journal. Ce paramètre est facultatif. Il prend la valeur **False** par défaut. Ce paramètre, s'il est mis à True, peut générer un fichier journal volumineux. A n'utiliser que pour le débogage.

[LogComm = True]

Booléen. Indique si les commandes autres que DATA sont enregistrées dans le fichier journal. Ce paramètre est facultatif et vaut **True** par défaut.

[FichJnl =
"C:\Temp\SMTP_SRV.LOG"]

Chaîne. Chemin et nom du fichier journal. Si ce paramètre n'est pas fourni, il prend la valeur "C:\Temp\SMTP_SRV.LOG". Si ce paramètre est une chaîne vide (""), le journal ne sera pas écrit sur disque, mais seulement écrit dans la variable d'état.

Si la spécification de fichier fournie est invalide, **Environ\$("Temp")\SMTP_SRV.LOG** est utilisé à la place.

Le fichier journal **n'est pas remis à zéro**. Les nouvelles lignes sont toujours ajoutées à la fin. Pensez à vérifier sa taille de temps à autre. Vous pouvez l'effacer même si le serveur est en fonctionnement. Il sera recréé lors de l'écriture suivante.

[EnvoiQuitte = True]

Booléen. Si True, le serveur scrute la table, puis se ferme après la scrutation (et l'envoi des messages éventuellement en attente). Si False, le serveur se remet en attente après la scrutation. Ce paramètre est facultatif et vaut **True** par défaut.

Ce cas de figure correspond à un scénario d'envoi unique : création d'un ou plusieurs messages – envoi immédiat – arrêt du serveur.

En cas d'échec de l'envoi, le message reste en boîte d'envoi, mais ne sera pas expédié avant le démarrage suivant du serveur.

Si le message est important, il est préférable de mettre ce paramètre à **False**. Le formulaire restera chargé et scrutera régulièrement la table, selon la valeur définie par le paramètre *DelaiVerif* ci-dessous.

[DelaiVerif = 30]

Entier. Délai, en minutes, entre les scrutations de la table BoiteMail. Ce paramètre est facultatif et est réglé à **30 mn** par défaut.

Note : Lors de l'utilisation de la commande **(E)SMTP Lance**, le formulaire effectue sa première scrutation au bout de 0,5 secondes. Il se cale ensuite sur le délai souhaité.

Tout délai inférieur à 5 minutes sera ramené à cette valeur.

[DelaiReponse = 300]

Entier. Délai, en secondes, accordé au serveur SMTP distant pour répondre à une commande envoyée par libMAIL. Ce paramètre est facultatif. Sa valeur par défaut est de **300 s (5 mn)**.

Un délai inférieur à 60 secondes sera ramené à cette valeur.

Exemple

' Démarrer le serveur SMTP et envoyer

Call SMTPLance("smtp.domaine.fr", "domaine.fr", False, True, , False, 10)

6.2.4 Options SMTP étendues

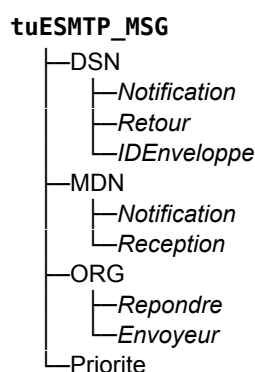
Les commandes de création de message (**CreeMail**) et de démarrage du serveur (**SMTPLance**) existent sous deux formes, la seconde permettant de spécifier des options étendues à transmettre au serveur distant (respectivement **ECreeMail** et **ESMTPLance**).

Remarque

Si vous fournissez des paramètres étendus à ces commandes alors que le serveur distant ne prend pas en charge l'extension idoine, ils seront simplement ignorés.

6.2.4.1 Options de message

Ces options étendues sont propres à un message, et peuvent être différentes d'un message à un autre. Elle sont transmises à la commande **ECreeMail** à l'aide d'une variable de type `tuSMTP_MSG`, dont voici la structure.



6.2.4.1.1 Avis de remise (DSN)

Le membre **DSN (Delivery Success Notification)** permet de passer les options pour les **avis de remise**. Ces avis sont émis par un serveur SMTP (le MTA - http://fr.wikipedia.org/wiki/Mail_Transfer_Agent) et informent l'émetteur sur le succès ou l'échec de la remise du message. L'avis de remise prend la forme d'un courriel émis par le serveur qui n'a pu faire suivre ou remettre le message dans la boîte aux lettres du destinataire.

À ce stade, le destinataire n'a pas encore démarré son client de messagerie pour récupérer son courrier. Il ne faut pas confondre les avis de remise avec les accusés de réception/lecture, qui sont envoyés par le **client de messagerie** lui-même, traités dans le paragraphe suivant.

DSN	Notification	Indique dans quel cas le serveur distant doit envoyer un avis de remise. Les valeurs possibles sont détaillées dans le tableau suivant. Ces valeurs peuvent être combinées (à l'aide de l'opérateur OR) pour obtenir un avis de remise dans plusieurs cas de figures.
	Retour	Détermine si l'avis de remise contient uniquement l'en-tête du message, ou le message complet. Voir tableau suivant.
	IDEnvelope	Permet d'ajouter un identifiant à l'avis de remise. La longueur maximale de l'identifiant est de 94 caractères. Il sera tronqué s'il dépasse cette longueur. L>IDEnvelope est ignoré si Retour prend la valeur <i>ImESMTPDsnRetDefault</i> . L>IDEnvelope doit être constituée uniquement de caractères dont le code ASCII est compris entre 33 et 126 inclus. Tout autre caractère sera supprimé automatiquement par libMAIL au moment de l'envoi.

Le membre **DSN** peut être initialisé à l'aide des constantes suivantes :

Notification	<i>ImIESMTPDsnNotifDefault</i>	Aucun paramètre n'est ajouté à la commande RCPT. Le serveur distant utilise ses réglages par défaut. Cette valeur est la valeur par défaut.
	<i>ImIESMTPDsnNotifJamais</i>	Demande au serveur distant de ne jamais envoyer d'avis de remise, quel que soit le résultat de l'envoi.
	<i>ImIESMTPDsnNotifSucces</i>	Demande au serveur distant d'envoyer un avis de remise si la remise du message s'effectue avec succès. Si l'envoi n'aboutit pas, aucune notification ne sera envoyée.
	<i>ImIESMTPDsnNotifEchec</i>	Demande au serveur distant d'envoyer un avis de remise si la remise du message échoue.
	<i>ImIESMTPDsnNotifDelai</i>	Demande au serveur distant d'envoyer un avis de remise si la remise du message est différée. Un serveur SMTP peut retenter l'envoi pendant un délai pouvant aller jusqu'à 5 jours.
	<i>ImIESMTPDsnNotifTous</i>	Cette constante est une combinaison des trois précédentes (<i>ImIESMTPDsnNotifSucces</i> OR <i>ImIESMTPDsnNotifEchec</i> OR <i>ImIESMTPDsnNotifDelai</i>). Demande au serveur distant d'envoyer une confirmation quel que soit le résultat de l'envoi.
Retour	<i>ImIESMTPDsnRetDefault</i>	Aucun paramètre n'est ajouté à la commande MAIL. Le serveur distant utilise ses réglages par défaut. IDEnvelope est ignoré. Cette valeur est la valeur par défaut.
	<i>ImIESMTPDsnRetHdrs</i>	Le serveur distant n'inclut que l'en-tête du message dans l'avis de remise .
	<i>ImIESMTPDsnRetFull</i>	Le serveur distant joint le message d'origine, avec ses éventuelles pièces jointes, à l'avis de remise .

Les valeurs *ImIESMTPDsnNotifJamais* et *ImIESMTPDsnNotifSucces-ImIESMTPDsnNotifEchec-ImIESMTPDsnNotifDelai* s'excluent mutuellement. Si *ImIESMTPDsnNotifJamais* est utilisée conjointement avec une des autres constantes, c'est *ImIESMTPDsnNotifJamais* qui prime.

6.2.4.1.2 Accusés de réception et de lecture (MDN)

Les accusés de réception et de lecture sont générés par les clients de messagerie (MS-Outlook, Thunderbird, etc...) eux-mêmes. Ils permettent de savoir que le message est bien arrivé sur le poste du destinataire. Ils prennent la forme d'un courriel émis par son client de messagerie. Ils sont décrits par la RFC 3798.

Ces demandes de notifications ne sont pas fiables : le client de messagerie peut être configuré pour les ignorer, automatiquement ou sur action de l'utilisateur.

Le membre MDN de tuESMTP_MSG permet de transmettre les informations nécessaires.

MDN	Notification	<i>Disposition-Notification-To</i> : Adresse de messagerie à laquelle il faut retourner un avis de lecture (ou de suppression sans lecture) du message. Le client de messagerie ou l'utilisateur peuvent choisir de ne pas répondre à la demande de notification. Cette option est décrite par la RFC 3798.
	Reception	<i>Return-Receipt-To</i> : Adresse de messagerie à laquelle il faut retourner un accusé de réception du message. Le client de messagerie ou l'utilisateur peuvent choisir de ne pas répondre à la demande de notification. Cette option n'est définie par aucune RFC, mais la plupart des clients de messagerie sont capables de la traiter.

6.2.4.1.3 Champs Origine

Les champs d'origine du message (chapitre 3.6.2 de la RFC 5322) permettent d'ajouter une adresse de réponse au message (si elle est différente de l'adresse de l'expéditeur, ainsi que l'envoi d'un mail au nom d'une autre personne. Elle peuvent être initialisées à l'aide du membre ORG de la variable d'options de message.

ORG	Repondre	<i>Reply-To</i> : Adresse de messagerie utilisée par le client de messagerie lorsqu'on clique sur le bouton Répondre. Ce champ peut contenir 0, 1 ou plusieurs adresses, séparées par des point-virgules.
	Envoyeur	<i>Sender</i> : ce champ permet à un utilisateur d'envoyer un message au nom d'un autre. <i>Exemple extrait de la RFC 5322 :</i> Une secrétaire doit envoyer un message pour une autre personne, la boîte à lettre de la secrétaire apparaîtra dans le "Sender:" et le nom de l'auteur réel apparaîtra dans le champ "From:" Certains clients de messagerie peuvent afficher « de adresse Sender de la part de adresse From » pour préciser le fait que l'envoyeur et l'auteur du message sont deux personnes distinctes. Cette adresse doit être différente de l'adresse de l'expéditeur. S'il n'y a qu'une seule adresse d'expéditeur, et que l'expéditeur est identique à l'envoyeur, ce dernier ne sera pas ajouté aux en-têtes du message.

6.2.4.1.4 Priorité du message

Le membre Priorite de tuESMTP_MSG permet d'attribuer une importance au message. **Cette option n'a aucune influence sur la manière dont les serveurs SMTP traitent le message (il ne sera pas transmis plus rapidement s'il a une priorité élevée).** Elle n'est utilisée que par les clients de messagerie, pour l'affichage de l'indicateur de priorité du message.

Priorite	Ce membre peut être initialisé à l'aide de l'une des constantes <i>ImlMsgPrioBas</i> , <i>ImlMsgPrioNrm</i> ou <i>ImlMsgPrioHte</i> . S'il n'est pas initialisé, l'option X-Priority n'est pas ajoutée à l'en-tête du message.
----------	---

6.2.4.2 Options de Serveur

Ces options sont utilisées par la commande **ESMTP Lance**, et restent actives tant que le serveur n'est pas relancé avec de nouvelles options.

Elles se présentent sous la forme d'une donnée de type utilisateur (tuESMTP). Il faudra créer une variable de ce type, l'initialiser et la passer à la procédure.

```
tuESMTP
└─AUTH
    ├──Methode
    ├──Utilisateur
    └─MotDePasse
```

Les informations relatives à l'**authentification** sont regroupées dans le membre **AUTH**.

AUTH	Methode	Méthode d'authentification. Voir le tableau ci-dessous pour les méthodes
------	---------	--

		autorisées. Si la méthode demandée n'est pas prise en charge par le serveur distant, libMAIL tente d'envoyer le message sans s'être authentifié.
	Utilisateur	Identifiant de l'utilisateur pour la connexion avec authentification.
	MotDePasse	Mot de passe de l'utilisateur.

Remarque

Les informations **Utilisateur** et **MotDePasse** ne sont pas inscrites dans le fichier journal. Elles sont remplacées par une chaîne de caractères banalisée.

Les valeurs possibles pour le membre **Methode** sont :

<i>lmESMTPAuthAucune</i>	Connexion sans authentification. Les membres Utilisateur et MotDePasse seront ignorés. Cette valeur est la valeur par défaut.	
<i>lmESMTPAuthLogin</i>	L'authentification utilise la méthode LOGIN. Utilisateur et mot de passe sont requis. C'est souvent l'adresse de messagerie complète, ou l'adresse sans le domaine (sans l'@ et tout ce qui suit). L'identifiant et le mot de passe sont envoyés séparément, simplement encodés en Base64.	
<i>lmESMTPAuthPlain</i>	L'authentification utilise la méthode PLAIN. Utilisateur et mot de passe sont requis. Ils sont envoyés simultanément, simplement encodés en Base64.	
<i>lmESMTPAuthCRAMMD5</i>	L'authentification se fait à l'aide de la méthode CRAM-MD5.	Le mot de passe est utilisé pour créer une signature dont l'empreinte MD5 est retournée au serveur. Le mot de passe n'est pas envoyé directement au serveur et ne peut donc pas être intercepté.
<i>lmESMTPAuthDIGESTMD5</i>	L'authentification se fait à l'aide de la méthode DIGEST-MD5.	

Exemple

Voici, à l'aide d'un exemple, la préparation à effectuer pour passer des paramètres étendus à **ESMTPLance**.

```
...
Dim tuOptions As tuSMTP
...
With tuOptions.AUTH
    .Methode = lmESMTPAuthLogin
    .Utilisateur = "identifiant"
    .MotDePasse = "motdepasse"
End With

Call ESMTPLance("smtp.domaine.fr", tuOptions, , False, True, , False, 10)
...
```

6.2.5 Obtenir l'état du serveur

La version 1.30 de la bibliothèque introduit une interface permettant à une application d'obtenir aisément l'état interne du serveur. À partir de cette version, le formulaire frm_SMTP, qui est le cœur du serveur, n'est plus affiché. Les informations qu'il fournissait, qui se limitaient à la visualisation du journal de connexion, ont été complétées et rendues accessibles à l'application principale.

La communication avec l'utilisateur se fait maintenant par l'intermédiaire de la zone de notifications de Windows : une icône reflète l'état du serveur. Un certain nombre de messages et d'informations à destination de l'utilisateur transitent également par cette icône.

La variable d'état interne est accessible par code, ce qui permet à une application de connaître l'état du serveur.

Le menu contextuel accessible par un clic droit sur l'icône permet également à l'utilisateur d'interagir avec le serveur SMTP (voir page 42).

6.2.5.1 La variable d'état

Cette variable permet à une application d'obtenir l'état du serveur par l'intermédiaire d'une nouvelle fonction associée à une structure de données.

Au cours de son fonctionnement, le serveur tient à jour une variable interne, reflet de son état à l'instant t . Cette variable est accessible en lecture seule (elle est locale à la bibliothèque, et ne peut pas être manipulée directement) par l'intermédiaire de la fonction **SMTPEtatSrv**. La fonction ne nécessite pas de paramètre, et retourne une valeur de type `tuEtatSrv`.

La variable interne n'est pas réinitialisée à la fin de l'envoi, et peut donc être interrogée par la suite. Elle est remise à zéro lors de la scrutation suivante, que celle-ci soit déclenchée par l'appel d'une procédure de démarrage (**(E)SMTPLance**, **SMTPRElance** ou **SMTPEnvoieMaintenant**) ou par l'écoulement du délai de scrutation.

Le résultat de la fonction peut être affecté à une variable de type `tuEtatSrv`, ou bien utilisé directement, sans affectation préalable.

Le tableau suivant récapitule les données d'état fournies par le serveur.

Etat	Octet	État du serveur. Voir le tableau ci-dessous pour les différentes valeurs possibles.
MessageEnCours	Entier	Position du traitement dans la liste des messages à envoyer.
MessagesTotal	Entier	Nombre total de messages en état 'E' dans la table au début du traitement.
OctetsEnvoyes	Entier long	Volume de données déjà envoyé, en octets. Cette valeur est calculée pour les en-têtes et les corps de messages proprement dits. Les données transmises lors de l'établissement et la fermeture de connexion ne sont pas comptabilisées.
OctetsTotal	Entier long	Volume total à envoyer, tous messages confondus, en octets. Même remarque que ci-dessus.
EnvoiDebut	Date/heure	Date et heure de début d'envoi du message proprement dit (hors établissement et fermeture de connexion).
EnvoiFin	Date/heure	Date et heure de fin de l'envoi (hors établissement et fermeture de connexion).
Resultat	Entier	Valeur numérique indiquant le résultat de l'envoi.
ScrutSvte	Date/heure	Date et heure de la scrutation suivante. Ce membre est remis à zéro lorsque le serveur est déchargé ou suspendu.

Voir le tableau détaillé page 34 pour plus d'informations sur les valeurs que peuvent prendre les différents membres.

À la fin de l'envoi (lorsque `Resultat` est différent de zéro), `MessageEnCours` est égal à `MessagesTotal` si tous les messages ont été transmis avec succès. Si `MessageEnCours` est inférieur à `MessagesTotal`, un ou plusieurs messages n'ont pu être transmis. Ces messages sont encore en état 'E' dans la table `BoiteMail` si l'erreur n'était que temporaire. Ils pourront alors être envoyés à nouveau. En cas d'erreur permanente, les messages concernés sont en état 'X' (invalidé).

Le membre `Etat` peut prendre l'une des valeurs suivantes :

État	Constante	Description
Déchargé	<i>ImISrvDecharge</i>	Le formulaire <code>frm_SMTP</code> n'est pas chargé. Le serveur est inactif. L'icône est absente de la zone de notifications.
Suspendu	<i>ImISrvSuspendu</i>	Le formulaire <code>frm_SMTP</code> est chargé, mais le serveur est suspendu. Aucune scrutation n'aura lieu.
En attente	<i>ImISrvAttente</i>	Le formulaire <code>frm_SMTP</code> attend la fin du délai avant la prochaine scrutation.
Connexion	<i>ImISrvConnexion</i>	Le formulaire <code>frm_SMTP</code> tente d'établir la connexion avec le serveur SMTP distant.
Envoi en cours	<i>ImISrvEnCours</i>	Le formulaire <code>frm_SMTP</code> est en train d'envoyer des messages.

État	Constante	Description
Annulation	<i>ImISrvAnnulation</i>	Une demande d'annulation de la transmission a été prise en compte.
Exécution d'une commande	<i>ImIExecCmd</i>	Cet état est transitoire. Il marque le temps entre l'appel d'une commande de démarrage ((E)SMTP Lance et SMTP Envoie Maintenant) et le démarrage effectif du Timer.

Exemple

Le code suivant attend la fin de l'envoi et affiche un message en fonction du résultat :

```

Call SMTPLance(...)

' Attend la fin de l'envoi et affiche un message
DoCmd.Hourglass True
Do While SMTPEtatSrv.Resultat = ImISrvResND
    DoEvents
Loop
DoCmd.Hourglass False

Select Case SMTPEtatSrv.Resultat
    Case ImISrvResOK
        MsgBox "Le message a été envoyé avec succès.", vbInformation, "Envoi d'un e-mail"

    Case ImISrvResErr
        MsgBox "Erreur lors de l'envoi du message.", vbExclamation, "Envoi d'un e-mail"
End Select

```

Remarque

Avant le premier démarrage du serveur, SMTPEtatSrv.Resultat vaut zéro. La boucle Do While...Loop ci-dessus ne se terminerait donc jamais si elle était exécutée avant l'appel à SMTPLance.

6.2.5.2 Contenu de la variable

Les tableaux suivants résument le contenu de la variable d'état en fonction de l'état du serveur.

États évolutifs, pendant les traitements.	Etat (Octet)	MessagesTotal (Entier)	MessageEnCours (Entier)	OctetsTotal (Entier long)	OctetsEnvoyes (Entier long)	EnvoiDebut (DateHeure)	EnvoiFin (DateHeure)	Resultat (Entier)	ScrutSvte (DateHeure)
Pendant le délai de démarrage du serveur.	<i>ImISrvExecCmd</i>	0	0	0	0	0	0	<i>ImISrvResND</i>	0
Phase de connexion.	<i>ImISrvConnexion</i>	n	0	0	0	0	0	<i>ImISrvResND</i>	0
En cours d'envoi.	<i>ImISrvEnCours</i>	n	$0 \leq x \leq n$	nn	$0 \leq y \leq nn$	Date/heure de début	0	<i>ImISrvResND</i>	0

États statiques, après la fin de la scrutation. La variable conserve cet état jusqu'au début de la scrutation suivante, moment où elle est remise à zéro.	Etat (Octet)	MessagesTotal (Entier)	MessageEnCours (Entier)	OctetsTotal (Entier long)	OctetsEnvoyes (Entier long)	EnvoiDebut (DateHeure)	EnvoiFin (DateHeure)	Resultat (Entier)	ScrutSvte (DateHeure)
Il n'y avait rien à envoyer.	<i>ImISrvDecharge</i> ou <i>ImISrvAttente</i> , selon le cas.	0	0	0	0	0	0	<i>ImISrvResRien</i>	0 ou date/heure, selon le cas.
Connexion impossible. Annulation pendant la connexion.		n	0	0	0	0	0	<i>ImISrvResCnx</i>	
Fin de l'envoi. Tous les messages ont été envoyés.			n	nn	nn	Date/heure de début	Date/heure de fin	<i>ImISrvResOK</i>	
Fin de l'envoi. Certains messages n'ont pas été envoyés.			$0 \leq x < n$		$0 \leq y < nn$			<i>ImISrvResErr</i>	

6.2.6 Contrôler le serveur

Une application peut faire appel aux commandes suivantes pour contrôler le serveur SMTP.

SMTPAnnule()

Permet d'annuler l'envoi en cours.

Le protocole SMTP ne permet pas d'annuler proprement la transmission d'un message lorsque sa partie DATA est entamée. Le message en cours d'envoi est donc transmis intégralement. Les messages suivants ne seront pas transmis. Ils restent dans la boîte d'envoi (état 'E'). Cette commande n'a aucun effet si le serveur n'est pas dans l'état **En cours** ou **Connexion**.

S'il n'y avait qu'un seul message en cours de traitement, il sera transmis normalement.

Si la demande d'annulation survient pendant la phase de connexion (état *ImISrvConnexion*), libMAIL ferme la connexion sans envoyer de message.

SMTPChange(*[NomSrv[:Port]]*, *[HELOdomain]*, *[LogData]*, *[LogComm]*, *[FichJnl]*, *[EnvoiQuitte]*, *[DelaiVerif]*, *[DelaiReponse]*)

Permet de modifier individuellement les différents paramètres du serveur. Elle reprend les mêmes éléments que la commande **SMTPLance**, mais contrairement à cette dernière, tous les arguments sont optionnels et aucune valeur par défaut n'est appliquée pour un paramètre manquant. Si un élément n'est pas fourni, sa valeur reste inchangée.

Reportez-vous à la description de **SMTPLance** (page 26) pour plus d'informations sur les différents paramètres.

VBA ne permet pas d'avoir un paramètre « Type Utilisateur » optionnel. Cette commande ne permet donc pas la modification des options passées à l'aide de *OptionsESMTP*.

Cette commande n'est effective que lorsque le serveur est dans l'état *ImISrvAttente* ou *ImISrvSuspendu*. Elle est ignorée dans les autres cas. Elle **ne charge pas** le formulaire frm_SMTP, et de ce fait ne peut pas se substituer à **(E)SMTPLance**.

VBA offre la possibilité d'utiliser des arguments nommés. Vous pouvez utiliser cette syntaxe plutôt que d'insérer des virgules pour les positions non utilisées lorsque vous ne changez que quelques valeurs. Les arguments nommés peuvent apparaître dans un ordre quelconque.

Exemple

```
Call SMTPChange(EnvoiQuitte:=False, LogComm:=False)
```

SMTPDecharge()

Arrête le serveur SMTP et décharge le formulaire. Le déchargement du formulaire entraîne la disparition de l'icône de la zone de notifications.

Il faudra faire appel à la commande **(E)SMTPLance** pour le redémarrer. Cette commande n'est prise en compte que lorsque le serveur est dans l'état **Suspendu** ou **En attente**. Elle est ignorée dans les autres cas.

SMTPEnvoieMaintenant()

L'appel à cette procédure permet de déclencher une scrutation immédiate. Si des messages sont en attente dans la table, ils sont transmis. Cette commande est ignorée si le serveur n'est pas dans l'état **En Attente**.

Note : Cette commande effectue une RàZ de la variable d'état du serveur.

SMTPRelance(*[NouveauDelai]*)

Relance la scrutation de la table BoiteMail, en appliquant le nouveau délai de scrutation (en minutes), s'il est fourni. Sinon, c'est le délai précédent qui est rétabli.

- Si le serveur est **Suspendu**, la commande applique le nouveau délai ou rétablit le délai précédent, puis relance la scrutation.

- Si le serveur est **En attente**, la commande applique simplement le nouveau délai. Si aucun délai n'est fourni, le délai est réinitialisé à sa valeur précédente.

La commande est sans effet dans les autres états.

Note : Cette commande effectue une RàZ de la variable d'état du serveur.

SMTPSuspend()

Suspend la scrutation de la table BoiteMail. Le formulaire n'est pas déchargé. La commande **SMTPRelance** reprend la scrutation. Cette commande n'est prise en compte que si le serveur est dans l'état **En attente** et n'a pas d'effet dans les autres cas.

Remarque

Ces commandes ne peuvent être utilisées qu'après un premier appel à **SMTPLance**. Dans le cas contraire, elles seront simplement ignorées.

Tableau récapitulatif des commandes et leur effet en fonction de l'état du serveur :

Commande	Etat	ImISrvDecharge	ImISrvSuspendu	ImISrvAttente	ImISrvConnexion ImISrvEnCours	ImISrvAnnulation
ESMTPLance(params)		- Charge le formulaire; - Applique les paramètres ; - Déclenche le timer au bout de 0,5 s.	- Modifie les paramètres ; - Déclenche le timer au bout de 0,5 s.		- Pas d'effet.	- Pas d'effet.
SMTPSuspend		- Pas d'effet.	- Pas d'effet.	- Suspend la scrutation en mettant le TimerInterval à 0.	- Pas d'effet..	- Pas d'effet.
SMTPRelance([Dela])		- Pas d'effet.	- Rétablit ou applique le nouveau délai de scrutation (TimerInterval).		- Pas d'effet.	- Pas d'effet.
SMTPAnnule		- Pas d'effet.	- Pas d'effet.	- Pas d'effet.	- Termine le message en cours d'envoi, puis met en attente. Pas d'envoi si l'annulation intervient pendant la connexion.	- Pas d'effet.
SMTPEnvoieMaintenant		- Pas d'effet.	- Pas d'effet.	- Démarre la scrutation immédiatement.	- Pas d'effet.	- Pas d'effet.
SMTPDecharge		- Pas d'effet.	- Décharge frm_SMTP.		- Pas d'effet.	- Pas d'effet.
SMTPChange		- Pas d'effet.	- Applique les nouveaux paramètres.		- Pas d'effet.	- Pas d'effet.

6.2.7 Limiter le menu

Par défaut, toutes les options du menu accessible par l'intermédiaire de l'icône de la zone de notifications sont actives, offrant à l'utilisateur le contrôle complet du serveur. Il est toutefois possible de restreindre les actions de l'utilisateur, à l'aide de la commande suivante :

EtatMenu(ValeurEtat)

ValeurEtat est une valeur de type Entier long, constituée par la combinaison d'une ou plusieurs des constantes suivantes :

Constante	Désactive l'option
<i>ImIMnuSspn</i>	Suspendre
<i>ImIMnuRInc</i>	Relancer
<i>ImIMnuEnvM</i>	Envoyer maintenant

Constante	Désactive l'option
<i>ImIMnuDech</i>	Décharger
<i>ImIMnuAnnE</i>	Annuler l'envoi
<i>ImIMnuNMsg</i>	Nouveau message
<i>ImIMnuGest</i>	Gestionnaire...
<i>ImIMnuEtat</i>	Afficher l'état
<i>ImIMnuAJnl</i>	Afficher le journal

Les constantes permettent de **désactiver** l'option correspondante du menu. Une option de menu désactivée par le serveur lui-même (en fonction du contexte) ne pourra pas être activée par cette commande.

Vous pouvez combiner plusieurs options à l'aide de l'opérateur OR.

Exemple

```
Call EtatMenu(lmIMnuGest OR lmIMnuNMsg)
```

Cette instruction désactive les options « Gestionnaire » et « Nouveau message »

6.2.8 Gérer le journal

Quatre commandes permettent la gestion du journal.

SMTPJournal()

Retourne une chaîne de caractères représentant le journal de connexion. Cette chaîne a une longueur maximale de 64000 caractères. Aucun paramètre n'est requis.

SMTPFormJnl()

Affiche le formulaire permettant de consulter le journal à l'écran. Seuls les 64000 derniers caractères sont conservés en mémoire. Le journal disque n'est pas limité en taille. C'est lui qu'il faudra consulter pour avoir un historique complet des connexions. Aucun paramètre n'est requis.

SMTPJnlRAZ([bDisque])

Permet d'effacer le journal. Sans paramètre, ou si *bDisque* vaut False, seule la variable est remise à blanc.

Si *bDisque* vaut True, le fichier disque est effacé également. La commande tente d'effacer le fichier disque correspondant à la valeur passée par l'intermédiaire des commandes **(E)SMTPLance**, **SMTPChange** et/ou **SMTPJnlFichier**. Si un premier fichier journal a été créé, puis qu'une des deux commandes précédentes a été utilisée pour spécifier un autre fichier journal (ou pas de fichier journal), **SMTPJnlRAZ** ne sera pas capable d'effacer le premier fichier créé, qui restera sur le disque dur.

SMTPJnlFichier([SpecFichier])

Cette procédure adopte le même mode de fonctionnement que le paramètre FichJnl de **(E)SMTPLance**. Elle permet d'initialiser rapidement la variable interne du serveur, sans démarrer le serveur lui-même. La bibliothèque n'utilise qu'un seul et même fichier journal, que son nom ait été passé par **(E)SMTPLance**, ou par **SMTPJnlFichier**.

Note : Il n'y a pas de journalisation vers le fichier tant que la variable interne de la bibliothèque n'a pas été initialisée par un appel à **(E)SMTPLance**, **SMTPChange** ou **SMTPJnlFichier**.

6.2.9 Commandes annexes

libMAIL expose quelques fonctions qui peuvent vous être utiles.

Enc_Base64(Chaine_de_caractères[, ICRLF])

Cette fonction retourne une chaîne de caractères, résultat de l'encodage en base 64 [2] de la chaîne passée en paramètre.

Un retour chariot est inséré dans le résultat tous les *ICRLF* caractères. *ICRLF* est ramené au multiple de 4 le plus proche, le cas échéant. Par défaut, la valeur de *ICRLF* est 76.

Lorsque *ICRLF* vaut zéro, aucun retour chariot n'est inséré.

Exemple

```
Debug.Print Enc_Base64("Chaîne De Caractères Quelconque")
```

Affichera : **Q2hhaW5lIERlIENhcmFjd0hyZXMgUXVlbGNvbnF1ZQ==** dans la fenêtre de débogage.

Pour 3 octets en entrée, nous obtenons 4 octets en sortie, avec seulement 6 bits utiles par octet. Vous comprenez maintenant pourquoi une pièce jointe occupe 1/3 d'espace en plus que le fichier d'origine...

Enc_Base64 a fait l'objet de soins particuliers concernant son optimisation [3], pour offrir des performances acceptables, même avec des pièces jointes de grande taille. VBA est un langage interprété, et donc pas spécialement véloce. Malgré cela, une chaîne de 3 millions de caractères peut être encodée en une seconde environ sur un PC récent (Core 2 Duo à 2,4 GHz).

Dec_Base64(Chaine_de_caractères)

C'est la fonction complémentaire de la précédente. Lorsqu'on lui fournit une chaîne encodée en Base 64, elle retourne la chaîne décodée.

Enc_QP(Chaine_de_caractères[, QEncoding])

Dec_QP(Chaine_de_caractères[, QEncoding])

Retourne la chaîne de caractères encodée en Quoted Printable, conformément à la RFC 2045. Dans cette représentation, tout caractère dont le code ASCII est supérieur à 127 est remplacé par son numéro ASCII en hexadécimal, précédé du signe égal (=).

Exemple

```
Debug.Print Enc_QP("Chaîne de caractères.")
```

Affichera : **Cha=E8Ene de caract=E8res.** dans la fenêtre de débogage.

```
Debug.Print Enc_QP(UaUTF8("Chaîne de caractères."))
```

Affichera : **Cha=C3=A8Ene de caract=C3=A8res.** dans la fenêtre de débogage.

Le paramètre facultatif *QEncoding* modifie le fonctionnement de **Enc_QP** pour permettre l'encodage du champ Objet et les noms des pièces jointes. Lorsqu'il vaut True, l'espace est remplacé par le caractère '_' et les caractères '?' et '_' sont également encodés (syntaxe *Encoded-Word* de la RFC 2047).

ExporteEML(ID[, Spécification_de_fichier])

Exporte le message d'identifiant ID vers un fichier au format eml. Tout message, quel que soit son état, peut être exporté à l'aide de cette fonction.

La spécification de fichier détermine l'emplacement et le nom du fichier de sauvegarde du message. Si elle n'est pas fournie, libMAIL crée un fichier <ID>.eml dans le dossier **Mes Documents**. L'extension du fichier créé par cette fonction sera toujours **eml**, indépendamment de ce que contenait la spécification d'origine.

La valeur de retour de la fonction est **zéro** si le fichier a été créé correctement, **-1** si aucun message ne correspondait à **ID**, ou tout autre code d'erreur VBA si la création a échoué.

Les fichiers eml peuvent être ouverts par la plupart des clients de messagerie actuels.

Si vous éprouvez des difficultés à l'ouvrir avec Outlook 2003, cet article de la base de connaissances Microsoft peut vous aider : <http://support.microsoft.com/kb/967346>

FrmEstCharge(*NomDeFormulaire*)

Retourne True ou False selon que le formulaire est chargé ou non (quel que soit son mode – normal, création, nouveau).

Joindre(*sTableau*, [*sDelim*])

Concatène les éléments du tableau *sTableau* en délimitant les éléments à l'aide du séparateur fourni dans *sDelim*. Si *sDelim* n'est pas fourni, l'espace (ASCII 32) est utilisé par défaut.

Cette fonction est l'équivalent de **Join** de VB6.

MD5(*Chaine_de_caractères*)

Retourne l'empreinte numérique de la chaîne de caractères passée en paramètre, calculée à l'aide l'algorithme de hachage MD5 (<http://fr.wikipedia.org/wiki/Md5>) décrit par la RFC 1321.

MD5 retourne une chaîne de 16 caractères où chaque caractère représente un octet de l'empreinte.

Pour obtenir la représentation hexadécimale de l'empreinte (une chaîne de 32 caractères hexadécimaux) appliquez la fonction **myHEX** au résultat de **MD5**.

Exemple

```
Debug.Print MD5("Chaîne de caractères.")
Retournera : _7İ_Ń_0hXCŋ_v_2İ

Debug.Print myHEX(MD5("Chaîne de caractères."))
Affichera : 0137cd81d113d8685843b61e761132ec
```

myComputerName()

Retourne une chaîne de caractères représentant le nom complet de l'ordinateur, avec le domaine s'il existe.

myCurrentUser()

Retourne le nom de l'utilisateur ayant ouvert la session Windows. Si la récupération de ce nom échoue, la fonction retourne le nom de l'utilisateur de base de données (**CurrentUser**). Si la sécurité n'est pas activée, ce sera probablement Admin.

NbMails(*[Différes]*)

Retourne le nombre de messages en attente d'envoi (nombre d'enregistrements en état 'E' dans la table BoiteMail).

Différes est un paramètre booléen optionnel. Par défaut, il vaut **False** et **NbMails** retourne le nombre de messages prêts à être envoyés. Les messages différés ne sont pas comptés.

Appelé avec **True**, **NbMails** retourne le nombre total de messages en état 'E'.

PJFichier(*Spécification_de_fichier*, [*NbCar*])

Cette fonction lit le fichier dont le chemin d'accès complet est passé en paramètre, et retourne une chaîne de caractères représentant le contenu de ce fichier.

Si *NbCar* est fourni, PJFichier ne retourne que les *NbCar* premiers caractères du fichier. *NbCar* vaut -1 par défaut, ce qui correspond à la totalité du fichier.

Les spécifications VBA limitent la taille d'une variable de type chaîne à environ 2 milliards de caractères, ce qui devrait être plus que suffisant pour traiter une pièce jointe.

Pour mémoire, beaucoup de fournisseurs d'accès limitent la taille d'un mail à 5 ou 10 Mo. Par respect pour le destinataire du mail, il est recommandé de garder les pièces jointes aussi réduites que possible (tout le monde ne dispose pas encore de débits très élevés...)

Purge(*n*|*#date*)

Purge la table BoiteMail. Cette fonction accepte un paramètre, numérique ou date.

Elle retourne le nombre d'enregistrements supprimés (de type entier long).

Purge(<i>n</i>)	Si $n > 0$, efface tous les mails en état 'V', sauf les n derniers. Si $n = 0$, tous les mails en état 'V' sont supprimés de la table. Si $n < 0$, conserve les n derniers jours, à partir de la date du jour.
Purge(<i>#Date</i>)	Efface tous les mails en état 'V' dont la date de création (DateMsg) est antérieure à Date.

Si le paramètre ne peut être évalué comme une valeur numérique ou date, la commande est sans effet.

Remarque

Purge supprime les enregistrements dans l'état 'V' dont la date de conservation est dépassée conformément au tableau ci-dessus, ainsi que **tous** les enregistrements 'D' (marqués pour suppression). Les enregistrements invalidés ('X') ne sont pas supprimés. Vous devrez les effacer manuellement.

Remplacer(*Chaine_de_caractères*, *sCherche*, *sRemplace*, [*IDebut*], [*INbRempl*], [*Compare*])

Similaire à la fonction **Replace**, introduite dans VB6 (Access 2000 et suivants).

Elle permet de remplacer les occurrences d'une chaîne de caractères dans une autre.

Élément

Description

Chaine_de_caractères **Chaîne.** Chaîne de caractères dans laquelle se fera le remplacement.

sCherche **Chaîne :** chaîne de caractères qui sera remplacée.

sRemplace **Chaîne :** chaîne de caractères qui remplacera la chaîne cherchée. Peut éventuellement être une chaîne vide (""). Dans ce cas, la chaîne cherchée est supprimée de la chaîne d'origine.

[*IDebut* = 1] **Chaîne.** La recherche commencera à la position fournie par IDebut. Ce paramètre est facultatif. Par défaut, la recherche commence au premier caractère de la chaîne de caractères d'entrée.

[*INbRempl* = -1] **Long.** Nombre de remplacements à effectuer. Ce paramètre est facultatif, et vaut -1 par défaut. Ceci équivaut à remplacer toutes les occurrences de *sCherche*. Dans le contraire, seules les *INbrempl* premières occurrences sont remplacées. Les occurrences suivantes ne sont pas remplacées.

[*Compare* = vbCompareMethod.vbBinaryCompare] **Entier.** Méthode de comparaison. Ce paramètre est facultatif et vaut *vbCompareMethod.vbBinaryCompare* par défaut.

Scinder(*Chaine_de_caractères*, [*Delimiteur*], [*NbFragments*], [*Compare*])

Similaire à la fonction **Split**, introduite dans VB6 (Access 2000 et suivants).

Elle sépare la chaîne de caractères fournie au niveau du délimiteur spécifié, et retourne un tableau constitué des sous-chaînes de la chaîne d'entrée.

Élément

Description

Chaine_de_caractères **Chaîne.** Chaîne de caractères à fractionner en sous-chaînes.

[*Delimiteur* = " "] **Chaîne.** Caractères qui délimitent les sous-chaînes à l'intérieur de la chaîne d'entrée. Ce paramètre est facultatif. Par défaut, la séparation se fera sur le caractère Espace.

[NbFragments = -1] **Entier.** Nombre de sous-chaînes à retourner.
 Ce paramètre est facultatif, et vaut -1 par défaut. Ceci équivaut à extraire toutes les sous-chaînes.
 Si ce paramètre est supérieur à zéro, la fonction retournera le nombre de sous-chaînes demandé.
 Par exemple, si NbFragment vaut 2, Scinder retournera un tableau comportant 2 lignes : la première ligne contiendra la première sous-chaîne, la seconde ligne tout le reste de la chaîne d'entrée.

[Compare = vbCompareMethod.vbBinaryCompare] **Entier.** Méthode de comparaison.
 Ce paramètre est facultatif et vaut *vbCompareMethod.vbBinaryCompare* par défaut.

SMTPTest(*nom.du.serveur[:port]*)

Cette commande effectue une connexion au serveur dont le nom est passé en paramètre, envoie une commande EHLO, puis referme la connexion.

Elle retourne ensuite la réponse du serveur.

Exemples

```
Debug.Print SMTPTest("smtp.free.fr")
```

Connexion à smtp.free.fr sur le port 25

```
220 smtp4-g21.free.fr ESMTP Postfix
--> EHLO vb-tests
250-smtp4-g21.free.fr
250-PIPELINING
250-SIZE 35000000
250-VRFY
250-ETRN
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
--> QUIT
221 2.0.0 Bye
```

```
Debug.Print SMTPTest("smtp.gmail.com:587")
```

Connexion à smtp.gmail.com sur le port 587

```
220 mx.google.com ESMTP t3sm33062100eeb.15
--> EHLO vb-tests
250-mx.google.com at your service, [82.65.60.80]
250-SIZE 35882577
250-8BITMIME
250-STARTTLS
250 ENHANCEDSTATUSCODES
--> QUIT
221 2.0.0 closing connection t3sm33062100eeb.15
```

UaUTF8(*Chaine_de_caractères*)

UTF8aU(*Chaine_de_caractères*)

Effectuent la conversion d'une chaîne de caractères Unicode vers une chaîne de caractères codée en UTF8 [7] et inversement. UTF-8 peut utiliser jusqu'à quatre octets pour encoder un caractère Unicode.

Exemple

```
Debug.Print UaUTF8("mélèze")
```

Affichera : mÃ©lÃ¨ze dans la fenêtre de débogage.

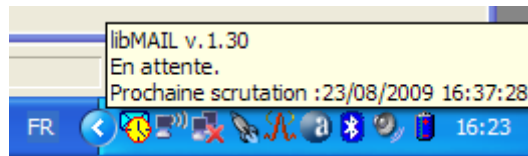
6.3 Interface utilisateur graphique

6.3.1 La zone de notifications

Lors du chargement de frm_SMTP, libMAIL affiche une icône dans la zone de notifications de Windows. Cette icône

donne une indication quant à l'état du serveur (voir tableau ci-dessous). Lorsque frm_SMTP est déchargé (à l'aide de la commande **SMTPDecharge**, par exemple), l'icône est retirée de la zone de notification, indiquant par là que le serveur n'est pas en fonctionnement.

Le chargement du serveur passe obligatoirement par un appel à la fonction **(E)SMTPLance** (voir page 26)



Les différents états de l'icône de notification :

Icône	Description
	Le serveur est suspendu, et ne reprendra pas la scrutation avant d'avoir été relancé par la commande (E)SMTPLance ou SMTPRelance . Aucun message n'est envoyé dans cet état.
	Le serveur attend que le délai entre deux scrutations soit écoulé.
	Établissement de la connexion avec le serveur distant.
	La connexion a été établie. Un ou plusieurs messages sont en cours d'envoi par le serveur.
	Une demande d'annulation de la transmission a été prise en compte.
	Le serveur a reçu une commande de démarrage ((E)SMTPLance, SMTPEnvoieMaintenant) et attend le démarrage effectif du Timer.

À la fin d'un envoi, cette icône affichera une notification indiquant que tous les messages ont été envoyés correctement, ou qu'il y a eu des erreurs. Le journal de connexion, s'il a été activé, donnera des informations détaillées sur les erreurs éventuelles.

6.3.2 Contrôler le serveur

L'icône de la zone de notifications est également le point d'entrée de l'interface utilisateur graphique. Un clic à l'aide du bouton droit de la souris permet à l'utilisateur de contrôler le fonctionnement du serveur par l'intermédiaire d'un menu.

	SMTPSuspend()
	SMTPRelance()
	SMTPEnvoieMaintenant()
	SMTPDecharge()
	SMTPAnnule()
	ECreeMailMIME()
	mnuGestMail()
	SMTPFormEtat()
	SMTPFormJnl()

Les différentes options de ce menu correspondent aux commandes vues précédemment. Reportez-vous au chapitre concerné pour obtenir de plus amples informations quant au fonctionnement de l'option.

Remarque

Access ne permet pas à une application d'afficher les barres de commandes créées dans une bibliothèque (à moins de la transformer en complément, ce qui compliquerait inutilement son installation). Pour cette raison, le

menu contextuel est créé directement dans l'application principale, sous forme de barre de commandes temporaire (elle est détruite automatiquement par Access lors de sa fermeture).
Le nom de la barre est **CB_libMAIL**.

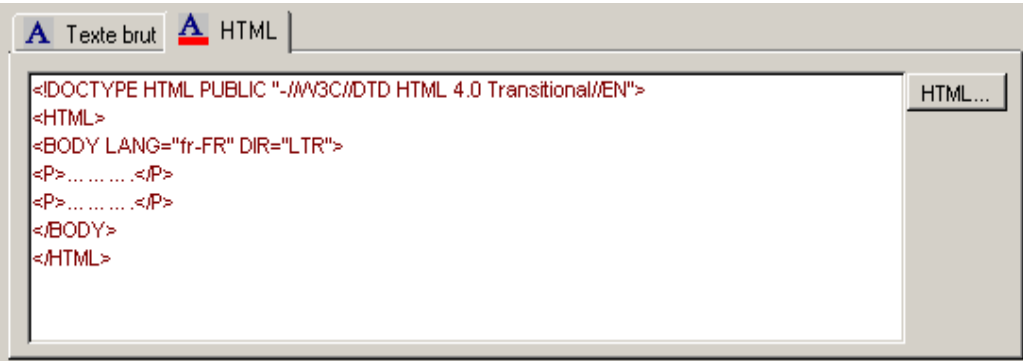
6.3.3 Édition interactive

Lorsque le paramètre *EditMail* de la commande **(E)CreeMail** est mis à True, libMAIL affiche un formulaire permettant de modifier le message avant de l'enregistrer dans la table BoiteMail. L'option **Nouveau message** du menu permet également d'afficher ce formulaire. Mais dans ce cas, tous les champs sont vides.

Si un ou plusieurs paramètres ont été fournis à **(E)CreeMail**, ils sont repris dans le formulaire. Vous pouvez les modifier librement avant d'envoyer le message.

Page principale :

Élément	Description
De	Adresse de messagerie de l'expéditeur. Elle est utilisée lors de la connexion au serveur distant et doit être connue de lui.
A, CC, CCI	Listes d'adresses de destinataires, séparées par des point-virgules. L'un de ces champs doit contenir au moins une adresse valide. Si le champ A est laissé vide, certains serveurs de messagerie peuvent rejeter le message ou renseigner le champ A avec la liste des destinataires CCI.
Objet	Objet du message.
Texte brut et HTML	Ces deux onglets contiennent le texte du message proprement dit. L'onglet Texte brut contient la version sans enrichissement du message. L'onglet HTML accueille une version enrichie du message, au format HTML. Vous pouvez y

Élément	Description
	<p>saisir ou coller un contenu HTML valide.</p>  <p>Vous pouvez aussi cliquer sur le bouton HTML, à droite de la zone de saisie, pour insérer le contenu d'un document HTML dans ce champ. La taille maximale de ce document est de 64000 octets. Il sera tronqué s'il dépasse cette taille.</p> <p>libMAIL générera automatiquement une version texte à partir de la version HTML si l'onglet texte brut n'est pas encore renseigné.</p>
Bouton Enregistrer	<p>Un clic sur ce bouton enregistre le message dans la table BoiteMail.</p> <p>Voir également envoyer immédiatement le message.</p>
Case envoyer immédiatement le message	<p>Si cette case est cochée, le serveur SMTP est démarré ou lancé immédiatement après l'enregistrement du message.</p> <p>L'envoi immédiat n'est possible que lorsque le serveur est dans l'état <i>Im/SrvDecharge</i> ou <i>Im/SrvAttente</i>.</p> <p>Tous les messages en boîte d'envoi (état 'E') sont alors envoyés, si l'état du serveur le permet à ce moment là.</p> <p>Voir également Onglet Serveur.</p>
Case Garder cette fenêtre ouverte	<p>Si cette case est cochée, le formulaire d'édition de message reste ouvert après le clic sur le bouton Enregistrer, permettant la saisie et l'envoi d'un nouveau message. Dans le cas contraire, le formulaire se ferme.</p>
Bouton RAZ	<p>Un clic sur ce bouton efface tous les champs du formulaire, vide la liste des pièces jointes et réinitialise les options à leur valeur par défaut. Seule la valeur du champ De est conservée.</p>
Bouton Fermer	<p>Ferme le formulaire d'édition de message.</p>
Onglet Fichiers joints	<p>Cet onglet liste les fichiers à joindre au message.</p> <p>La liste affiche le nom du fichier, sa taille en octets et son chemin d'accès. Le texte sous la liste vous informe sur le nombre de fichiers joints ainsi que sur la taille totale des pièces jointes.</p> <p>Un clic sur le bouton Ajouter affiche une boîte de dialogue permettant de sélectionner un ou plusieurs fichiers à partir d'un emplacement. Il est possible d'utiliser plusieurs fois ce bouton pour sélectionner des fichiers se trouvant dans des emplacements différents.</p> <p>Si un raccourci est choisi (fichiers .lnk), c'est bien le fichier cible qui sera joint au message, et non le raccourci lui-même.</p> <p>Le bouton Retirer permet de supprimer une ou plusieurs pièces jointes de la liste.</p>

Onglet Remise :

Cet onglet regroupe les options étendues propres aux messages. Vous pouvez vous reporter en page 28 pour une description complète de ces options.

Les choix faits sur cet onglet sont enregistrés avec le message, dans le champ ESMTP de la table BoiteMail.

L'option Jamais et le groupe Succès-Échec-Différé s'excluent mutuellement. Il est par contre possible de cocher zéro ou plusieurs options du groupe Succès-Échec-Différé. Lorsqu'aucune option de notification n'est cochée, c'est le comportement par défaut du serveur distant qui est utilisé.

Onglet Options

Sur cet onglet, vous pouvez saisir une adresse mail pour les accusés de réception et de lecture des messages. Le bouton, à droite de chaque zone de texte, permet d'insérer l'adresse de l'émetteur du message (contenu du champ **De** de la page principale).

L'option **Priorité** ne fait pas partie du protocole SMTP, et n'a aucune influence sur le traitement du message. Elle indique simplement au client de messagerie du destinataire qu'il faut marquer ce message avec une priorité donnée.

Les options **Envoyer le message après** et **Conserver le message jusqu'au** sont également propres à libMAIL. Elles permettent respectivement de différer l'envoi du message jusqu'à une date/heure définies et d'interdire la suppression du message par la fonction **Purge** avant une date et heure définies.

Onglet Serveur :

Les options de cet onglet ne s'appliquent que lors d'un envoi immédiat (lorsque la case **envoyer immédiatement le message** est cochée). Si la case n'est pas cochée, l'onglet est inactif.

Si la commande **(E)SMTP Lance** a été appelée auparavant avec des informations de serveur SMTP et éventuellement d'authentification, celles-ci sont reprises automatiquement dans les champs appropriés.

Si vous saisissez un serveur et/ou un login différent(s), ils ne seront utilisés que pour cet envoi. Les valeurs d'origines seront rétablies automatiquement dès la fin de l'envoi, même si les champs affichés par le formulaire conservent les

dernières valeurs saisies. Un envoi déarrant pendant que ce formulaire est ouvert utilisera les informations définies par la dernière commande **(E)SMTPLance** exécutée, et non celles saisies dans ce formulaire.

ATTENTION : ces informations ne sont pas conservées. Si l'envoi immédiat échoue, le message est toutefois enregistré dans la table, et il sera envoyé plus tard avec les valeurs de serveur/login actives au moment de cette nouvelle scrutation. Il se peut donc qu'un message créé à l'aide de ce formulaire soit acheminé vers le mauvais serveur distant.

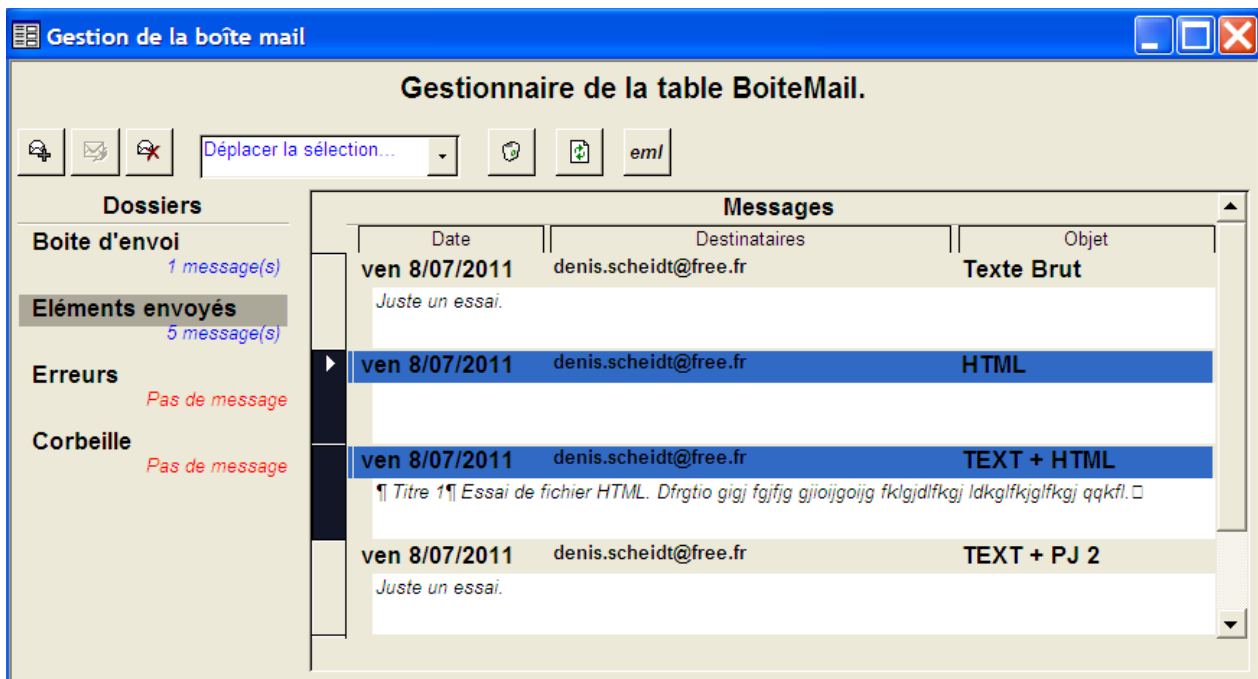
Les valeurs entrées dans ces champs servent à démarrer le formulaire serveur SMTP. Une description détaillée se trouve en page 30.

L'état précédent du serveur est préservé. S'il était en attente, il se remet en attente après l'envoi. S'il était déchargé, il se déchargera après l'envoi.

6.3.4 Gérer la table BoiteMail

Ce formulaire, appelé par l'intermédiaire du menu de l'icône de la barre des tâches, permet de gérer facilement les messages stockés dans la table BoiteMail.

Il autorise la création, la modification, le déplacement ou la suppression de messages, un peu à la manière d'un client de messagerie traditionnel.



La partie gauche du formulaire représente les différents états possibles pour les messages, sous forme d'une liste de dossiers.

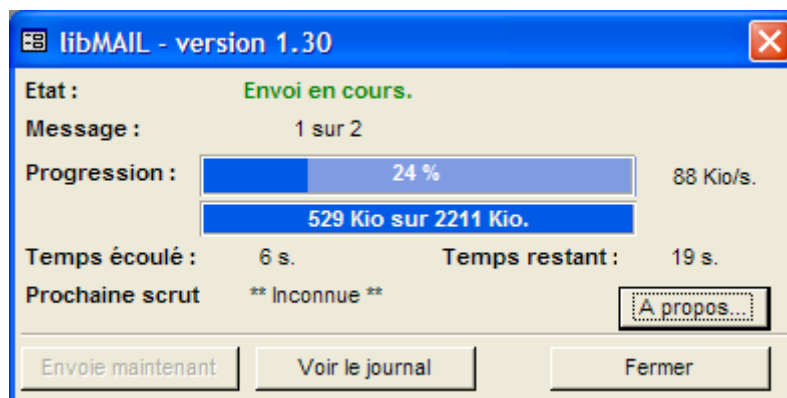
La partie droite liste les messages pour le dossier choisi.

Dans la partie supérieure du formulaire, différents boutons vont permettre d'agir sur les messages.

Icône	Description
	Création d'un message. Ce bouton appelle le formulaire d'édition de message.
	Modification d'un message. Un clic sur cette icône charge le contenu du message sélectionné dans le formulaire d'édition de message. Cette action n'est possible que dans la boîte d'envoi, et si le serveur n'est pas en train de transmettre les messages. Les messages des autres dossiers ne peuvent pas être modifiés.
	Envoie les messages sélectionnés vers la corbeille. La suppression depuis la boîte d'envoi n'est pas possible pendant que le serveur envoie les messages.
	Déplace les messages sélectionnés vers le dossier choisi. Il n'est pas possible de déplacer des messages depuis la boîte d'envoi pendant que le serveur expédie les messages.
	Vide le contenu de la corbeille.
	Actualise les dossiers et leur contenu.
	Exporte le message sélectionné vers un fichier .eml, pouvant être importé dans la plupart des clients de messagerie actuels.

6.3.5 Visualiser l'état du serveur

Les informations disponibles dans la variable interne du serveur peuvent être présentées à l'utilisateur grâce à un formulaire qui les affichera en temps réel. Ce formulaire va lire la variable d'état toutes les secondes, et afficher la progression de l'envoi. De ce fait, il peut avoir un peu de retard par rapport à l'icône de la zone de notifications.



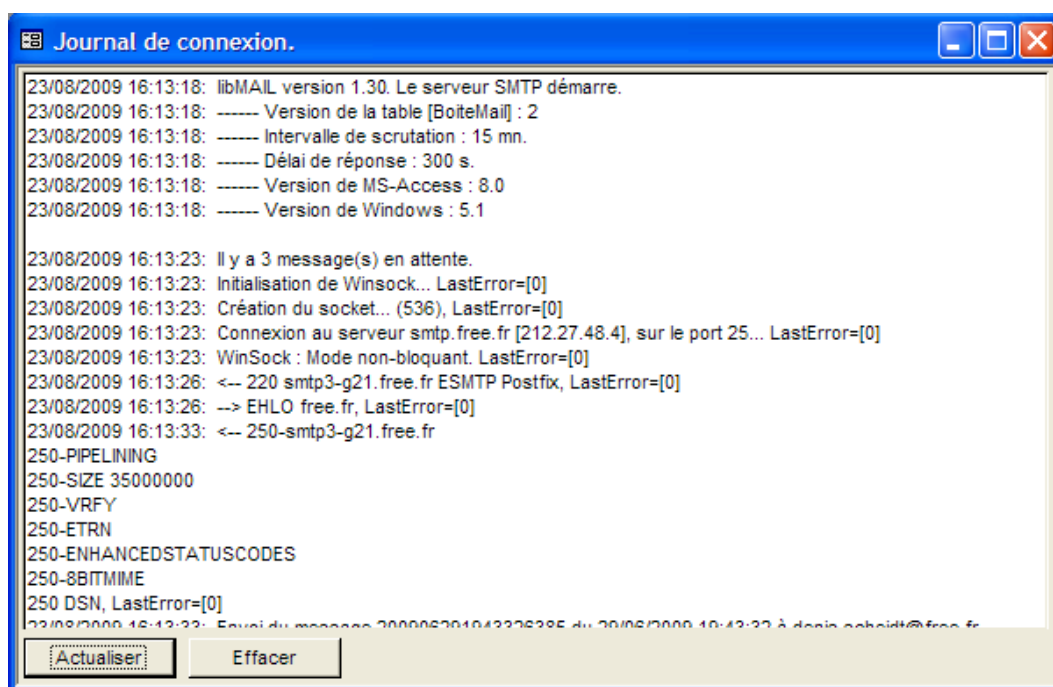
L'affichage et le masquage de ce formulaire sont contrôlés par la commande **SMTPFormEtat(bAffiche)**. *bAffiche* est une valeur de type booléen, True provoquant l'affichage du formulaire, False sa fermeture.

Lorsque le serveur est dans l'état **En attente**, un clic sur le bouton **Envoyer maintenant** déclenche une scrutation immédiate, sans attendre l'expiration du délai.

Un clic sur le bouton **Voir le journal** permet de consulter le journal de connexion.

6.3.6 Gérer le journal

Avant la version 1.30, le journal était affiché dans le formulaire frm_SMTP lui-même. Ce dernier étant maintenant invisible, l'affichage du journal est confié à un formulaire dédié, qui peut être appelé à partir du formulaire d'état, de l'icône de notification ou par code.



Le bouton **Effacer** ne réinitialise que la partie du journal qui réside en mémoire. Le fichier disque ne peut être effacé qu'à l'aide de la commande **SMTPJnIRAZ** avec *bDisque* = True.

7 Dépannage

Si libMAIL n'arrive pas à se connecter au serveur SMTP distant ou envoyer un message, la première chose à contrôler est le fichier journal qui se trouve par défaut dans **C:\Temp\SMTP_SRV.LOG**. Si le dossier C:\Temp n'existe pas, le journal est créé dans le dossier désigné par la variable d'environnement **TEMP** ou **TMP**. Ces variables désignent par défaut **C:\Documents and Settings\<NomProfil>\Local Settings\Temp**.

La lecture du journal peut donner des indications sur les causes de l'échec.

7.1 Contrôler la connexion

Si le journal indique un problème de connexion, voici quelques commandes de base qui vous permettront d'établir un premier diagnostic.

Ouvrez une invite de commande et tapez l'instruction suivante :

```
ping smtp.fai.fr
```

où vous remplacerez *smtp.fai.fr* par le nom réel du serveur SMTP auquel vous souhaitez vous connecter. Si tout est correct, vous devez obtenir une réponse ressemblant à ceci :

```
Envoi d'une requête 'ping' sur smtp.fai.fr [xxx.xxx.xxx.xxx] avec 32 octets de données :

Réponse de xxx.xxx.xxx.xxx : octets=32 temps<1ms TTL=128
Réponse de xxx.xxx.xxx.xxx : octets=32 temps<1ms TTL=128
Réponse de xxx.xxx.xxx.xxx : octets=32 temps<1ms TTL=128
Réponse de xxx.xxx.xxx.xxx : octets=32 temps=11 ms TTL=128

Statistiques Ping pour xxx.xxx.xxx.xxx:
    Paquets : envoyés = 4, reçus = 4, perdus = 0 (perte 0%),
    Durée approximative des boucles en millisecondes :
        Minimum = 0ms, Maximum = 11ms, Moyenne = 2ms

C:\>_
```

Nous savons maintenant que *smtp.fai.fr* est joignable depuis ce poste de travail. Certains routeurs sont configurés pour ne pas répondre au ping. Même si le test précédent est négatif, passez tout de même à l'étape suivante.

Le second test est destiné à vérifier que le serveur SMTP distant est bien actif. Tapez la commande suivante :

```
telnet smtp.fai.fr 25
```

Si votre serveur demande une connexion sur un port différent du port standard, remplacez le 25 de la ligne ci-dessus par la valeur correspondante.

Le serveur doit répondre avec un message commençant par 220. La suite du message est sans importance et varie d'un serveur à l'autre.

```
220 smtp.fai.fr [XMail 1.27 ESMTP Server] service ready
```

Tapez ensuite :

```
EHLO nimportequoi
```

Le serveur répond avec la liste des options étendues qu'il peut traiter :

```
250-smtp.fai.fr
250-VRFY
```

```
250-ETRN
250-8BITMIME
250-PIPELINING
250-AUTH LOGIN PLAIN CRAM-MD5
250-SIZE 7680000
250 STARTTLS
```

Pour clore la connexion, tapez :

```
QUIT
```

Le serveur distant ferme la connexion et vous revenez à l'invite de commande.

```
221 [XMail 1.27 ESMTP Server] service closing transmission channel

Perte de la connexion à l'hôte.

C:\>_
```

Remarque

À partir de la version 1.50 de libMAIL, vous pouvez utiliser la fonction `SMTPTest` dans la fenêtre d'exécution d'Access pour obtenir les mêmes informations :

```
? SMTPTest( "serveur.fai.fr")
Connexion à serveur.fai.fr sur le port 25
220 serveur.fai.fr ESMTP Postfix
--> EHLO nimportequoi
250-serveur.fai.fr
250-PIPELINING
250-SIZE 35000000
250-VRFY
250-ETRN
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
--> QUIT
221 2.0.0 Bye
```

```
? SMTPTest( "serveur.fai.fr:587")
Connexion à serveur.fai.fr sur le port 587
Impossible d'établir la connexion...
Erreur -1, socket 756
```

Si ceci ne vous permet pas de résoudre le problème, contactez-moi sur le forum (<http://www.self-access.com/forums/list.php?20>) ou par mail. Joignez à votre question la fin du journal (la partie correspondant au dernier ou deux derniers envois) ainsi que le résultat des commandes ci-dessus.

8 Problèmes potentiels

8.1 Tâche de fond

Lorsque le paramètre *EnvoiQuitte* de la commande **ESMTP Lance** est *False*, le formulaire *frm_SMTPLance* reste en mémoire et scrute la table *BoiteMail* à intervalles réguliers. Cette scrutation est déclenchée par le Timer du formulaire. La tâche d'envoi elle-même rend la main au système d'exploitation aussi souvent que possible, par l'intermédiaire de **DoEvents**. Cette construction permet d'obtenir un traitement en tâche de fond de l'envoi des messages.

Toutefois, Access ne permet pas l'utilisation de threads (ou fils d'exécution – http://fr.wikipedia.org/wiki/Thread_%28informatique%29), et ce traitement *pseudo-parallèle* ne peut fonctionner qu'à condition de prendre certaines précautions.

Lorsqu'on fait de la saisie dans un formulaire, Access passe la majorité de son temps à attendre une action de l'utilisateur. Dans ce cas de figure, la scrutation se déclenchera correctement.

Si un traitement entre dans une boucle de longue durée, par contre, il peut empêcher le déclenchement du Timer. Si cette boucle démarre pendant que le serveur traite un message, cela peut conduire à une interruption de l'envoi si la durée du blocage est supérieure au délai d'attente du serveur distant. Ce dernier, n'obtenant plus de réponse de la part de *libMAIL*, mettra fin à la connexion.

Ce dernier cas peut être évité en vérifiant l'état du serveur avant d'entrer dans la boucle :

```
' Si un envoi est en cours, en attendre la fin.  
Do While SMTPEtatSrv.Etat = lmlSrvEnCours  
    DoEvents  
Loop  
  
' Démarrer le traitement  
Do ...
```

Une alternative consiste à rendre la main au système d'exploitation de temps à autre, à l'aide de **DoEvents**. Cette solution peut introduire d'autres problèmes, l'utilisateur pouvant par exemple cliquer à nouveau sur le bouton qui a permis de lancer le traitement avant qu'il ne soit terminé, ou démarrer une autre tâche qui peut entrer en conflit avec la première. De plus, une utilisation intensive de **DoEvents** peut allonger considérablement la durée totale du traitement.

8.2 Figement de l'interface

Des cas de figement de l'interface Access ont été constatés lors de l'utilisation de certains logiciels antivirus. Le blocage se produit généralement au moment de l'envoi de la pièce jointe, plus précisément lors de l'émission du point (',') qui termine le corps du message. L'antivirus analyse le contenu du message à ce moment-là, et ne rend pas immédiatement la main à la routine d'envoi *EnvoiCMD*.

Le tableau suivant récapitule les anti-virus testés à ce jour :

Antivirus	Version	Résultat
Avast – Édition familiale <i>WXP-Pro SP3</i>	v4.8 v5.0	Avast intercepte et analyse la transmission. Pas de blocage notable.
Norman EndPoint Protection <i>WXP-Pro SP3</i>	v7.2	Blocage de l'application Access pendant l'analyse du corps du message. Désactiver l'analyse des messages sortants permet d'éviter le figement.
Trend Micro Worry-Free Business Security <i>WXP-Pro SP3</i>	v7.x	Intercepte et analyse la transmission. Pas de blocage notable.


Vous pouvez contribuer à la mise à jour de cette liste. Dites-moi si votre antivirus bloque Access ou pas :)

9 Contact

Si vous avez des questions, des suggestions, ou si avez découvert une erreur dans le code ou dans la documentation, ou juste pour me dire que ce bout de programme vous a été utile, vous pouvez m'envoyer un mail à access.libmail@gmail.com ou laisser un commentaire sur le forum libMAIL de Self-Access : <http://www.self-access.com/forums/list.php?20>

La licence LGPL vous donne le droit de modifier le code source de libMAIL. Si vous diffusez cette bibliothèque modifiée, les sources modifiées doivent être rendus accessibles.

Si vous pensez que ces modifications méritent de figurer dans libMAIL, vous pouvez me les faire parvenir à l'adresse ci-dessus, afin que je puisse les incorporer à la prochaine version.

Insérez simplement les caractères [libMAIL] dans l'objet de votre mail, ça me permettra de traiter plus rapidement votre message (ou cliquez ici : ).

Vous pourrez alors retrouver régulièrement une version mise à jour, incorporant les dernières améliorations, sur Self-Access.com (<http://www.self-access.com/cms/access/assistants/libmail>).

Si vous avez apprécié libMAIL, n'hésitez pas à encourager les développements futurs en cliquant sur le lien ci-dessous : [Faire un don](#)

Le montant du don est à la discrétion de chacun.

D'avance merci de votre participation.

10 Historique des modifications

- Novembre 2013 – 1.06 - Nouveautés et changements de la version 1.41.
- Juillet 2011 – 1.05
- Nouveautés et changements de la version 1.40.
 - Ajout des paragraphes 'Première utilisation', 'Commandes de base', 'Corps Texte et HTML' et 'Dépannage'.
 - Précision sur la priorité des constantes DSN.
 - Chapitre « Limiter le menu » déplacé sous « Interface de programmation ».
 - Ajout et modification de styles.
 - Refonte du paragraphe 'Fonctionnement du serveur' et de l'introduction du paragraphe 'Créer un courrier électronique'.
 - Chapitre 'Contact' : remplacement de l'adresse du forum principal par celle du forum dédié.
 - Mise en évidence des valeurs par défauts dans les tableaux descriptifs des paramètres de fonctions.
- Juin 2010 – 1.04
- Séparation de l'interface de programmation et de l'interface utilisateur.
 - Paragraphe 'Configuration requise'.
 - Paragraphe 'Nouveautés de la version'.
 - Nouvelles commandes et fonctions de la version 1.30.
 - Ajout de l'index.
 - Ajout des accusés de lecture/réception. Distinguo sur les avis de remise.
 - Coloration syntaxique à l'aide de COOder.
 - Paragraphe 'Problèmes potentiels'.
 - Tableau récapitulatif des commandes de contrôle.
- Juillet 2009 – 1.03
- Ajouts concernant la version 1.20 de libMAIL. Consultez le fichier Changelox.txt pour connaître la liste des modifications. Support de l'extension DSN (RFC 3798). Paragraphe concernant l'édition d'un message avant envoi.
 - Fonctions ErreursPJ Scinder, Enc_QP et Enc_UTF8.
 - Valeur de retour et retours d'erreurs de ECreeMail.
 - Description incorrecte d'EnvoiQuitte.
 - Valeur minimale de DelaiVerif.
 - Remarque concernant la bibliothèque DAO.
 - Correction d'erreurs typographiques.
 - Ajout des logos CC et LGPL.
- Mars 2009 – 1.02
- Ajouts concernant la version 1.10 de libMAIL. Consultez le fichier Changelox.txt pour connaître la liste des modifications. Commande ESMTPLance et constantes pour l'authentification.
 - Remarque sur la taille des messages.
 - Messages en état 'X'
 - Ajout de la liste des commandes SMTP prises et charge.
 - Précisions sur Purge et myCurrentUser
- Février 2009 – 1.01
- Ajouts concernant la version 1.01 de libMAIL. Consultez le fichier Changelog.txt pour connaître la liste des modifications.
 - Correction d'erreurs typographiques.
- Février 2009 – 1.00
- Version initiale.

11 Références

- [1] Les RFC traduites en français : <http://abcdrfc.free.fr/>
- [2] La conversion en base 64 : <http://fr.wikipedia.org/wiki/Base64>
- [3] Les performances des concaténations de chaînes : <http://support.microsoft.com/?scid=kb%3Ben-us%3B170964&x=19&y=13>
- [4] Encodage au format MIME : <http://fr.wikipedia.org/wiki/MIME>
- [5] La LGPL expliquée : http://wiki.venividilibri.org/index.php?title=GNU_Lesser_General_Public_License_Version_2.0
- [6] Traduction **non officielle** de la LGPL (v. 2.1) : http://www.linux-france.org/article/these/licence/lgpl/lgpl_monoblock.html
- [7] L'encodage en UTF-8 : <http://fr.wikipedia.org/wiki/UTF-8>
- [8] Les RFC sur le site de l'IETF : <http://www.ietf.org/rfc.html>

Index

A

accusés de réception.....	29
AUTH.....	30
Methode.....	31
avis de remise.....	28

B

base 64.....	38
BoiteMail.....	15

C

Configuration.....	13
CreeMail.....	18

D

Dec_Base64.....	38
Dec_QP.....	38
DSN.....	28
IDEnvelope.....	28
Notification.....	29
Retour.....	29

E

ECreeMail.....	18
ECreeMailMIME.....	18
Enc_Base64.....	38
Enc_QP.....	38
ErreursPJ.....	25
ESMTPLance.....	26
EtatMenu.....	36
ExporteEML.....	38

F

FrmEstCharge.....	39
-------------------	----

H

HTML.....	21
-----------	----

I

identifiant du message.....	24
-----------------------------	----

J

Join.....	39
Joindre.....	39
Journal.....	
SMTPFormJnl.....	37
SMTPJnlFichier.....	37
SMTPJnlRAZ.....	37
SMTPJournal.....	37

L

Licence.....	9
--------------	---

M

MD5.....	39
MDN.....	29
Menu.....	42
ModifieMail.....	26
myComputerName.....	39
myCurrentUser.....	39
myHEX.....	39

N

NbMails.....	39
Notification.....	
icône.....	42
zone.....	41

O

objet Access.....	23
ORG.....	30

P

PJFichier.....	39
PJOA.....	23

Priorite.....	30
Purge.....	40

Q

Quoted Printable.....	38
-----------------------	----

R

références.....	13
Remplacer.....	40
Replace.....	40

S

Scinder.....	40
SIZE.....	25
SMTPAnnule.....	35
SMTPChange.....	35
SMTPDecharge.....	35
SMTPEnvoieMaintenant.....	35
SMTPEtatSrv.....	32
SMTPFormEtat.....	48
SMTPLance.....	26
SMTPRelance.....	35
SMTPSuspend.....	36
SMTPTest.....	41
Split.....	40

T

table BoiteMail.....	15
tuEtatSrv.....	32

U

UaUTF8.....	41
UTF8aU.....	41

V

VerifieBAL.....	15
-----------------	----