Transférer Des Fichiers Volumineux avec Outlook

Sous-titre de l'article

Cet article vous permet d’installer des composants en VBA dans Outlook pour transférer des fichiers qui dépassent la limite admise par votre fournisseur de messagerie, sans avoir besoin de passer par un service externe de partage de fichiers.

Texte affiché avant le sommaire.

Multipage.

Texte de la licence de remplacement Référence de la licence de remplacement

Niveau du lecteur Durée de lecture

Liens supplémentaires.

Contenu

[I. Introduction 2](#_Toc59272502)

[II. Envoyer des fichiers volumineux avec Outlook 3](#_Toc59272503)

[III. Recevoir des fichiers volumineux avec Outlook 5](#_Toc59272504)

[IV. Installer (et désinstaller) TDFV dans Outlook – Automatiquement avec Excel 8](#_Toc59272505)

[V. Installer (et désinstaller) TDFV dans Outlook – Manuellement 9](#_Toc59272506)

[VI. Remarques sur le code VBA utilisé dans Outlook 13](#_Toc59272507)

[VII. Remarques sur le code VBA utilisé pour l’installation automatique avec Excel 17](#_Toc59272508)

[VIII. Conclusion 18](#_Toc59272509)

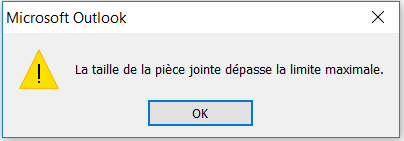
[IX. Fichiers Joints 18](#_Toc59272510)

[X. Annexe 1 – Compresser et décompresser un fichier avec Windows 19](#_Toc59272511)

[XI. Annexe 2 – Découper un fichier en lots et le recomposer 20](#_Toc59272512)

[XII. Remerciements 22](#_Toc59272513)

# Introduction



Comment envoyer par Outlook un fichier volumineux en pièce jointe lorsque sa taille dépasse la limite autorisée ?

Car comme indiqué dans la [documentation officielle](https://support.office.com/fr-fr/article/envoyer-des-fichiers-volumineux-avec-outlook-8c698842-b462-4a4c-8d53-5c5dd04f77ef) :

« *Outlook limite la taille des fichiers que vous pouvez envoyer. Cette limite empêche votre ordinateur de tenter continuellement de charger des pièces jointes très volumineuses qui dépassent les limites définies par la plupart des fournisseurs de services Internet. Pour un compte de courrier Internet tel que Outlook.com ou Gmail, la limite de taille cumulée des fichiers est de 20 mégaoctets (Mo), et, pour des comptes Exchange (courrier professionnel), la limite de taille cumulée des fichiers par défaut est de 10 Mo.*»

Certes, il existe des solutions de contournement (issues du même lien) :

« *Vous recevez un message d’erreur si vous tentez de joindre des fichiers d’une taille supérieure à la limite de taille maximale. Il existe plusieurs façons de contourner cette limite de taille, notamment en utilisant un service de partage de fichiers ou en compressant le fichier.*»

En résumé :

* Pour un fichier volumineux (c’est-à-dire un fichier qui dépasse la limite admise par votre fournisseur de messagerie), à défaut d’utiliser un service de partage de fichier, le plus simple est d’utiliser un programme de compression qui permet de découper un fichier en plusieurs « petits » fichiers (lots), par exemple avec 7-Zip, en indiquant « 10M » dans la partie « Diviser en volumes, octets » ce qui compresse le fichier source en plusieurs fichiers de 10 Mo.

Il ne vous restera plus qu’à joindre les fichiers en prenant soin de ne pas en oublier.

* Lorsque c’est l’ensemble des pièces jointes qui dépasse le quota d’un message, par exemple 20 Mo, il suffit d’envoyer les fichiers en plusieurs messages dont la taille respecte le quota admis, ici aussi en prenant soin de ne pas oublier de fichier.

Mais normalement avec ce niveau de manipulations, surtout s’il y a plusieurs fichiers volumineux à envoyer, vous devriez avoir perdu la majorité des utilisateurs de votre entreprise, quant à ceux qui vont réussir à se débrouiller ils trouveront l’exercice fastidieux et chronophage.

C’est pourquoi je vous propose dans cette documentation une solution plus simple et plus rapide, qui va réaliser automatiquement tous ces traitements de compression et de découpage des pièces jointes d’un message « hors quota », et les envoyer en un ou plusieurs messages avec Outlook.

Plus précisément, une application en VBA pour Outlook que j’ai nommée TDFV.

Seuls prérequis : disposer d’une version Outlook 2010 ou 2016 et être habilité(e) à activer les macros (normalement c’est le cas), et disposer du programme de compression 7-Zip.

# Envoyer des fichiers volumineux avec Outlook

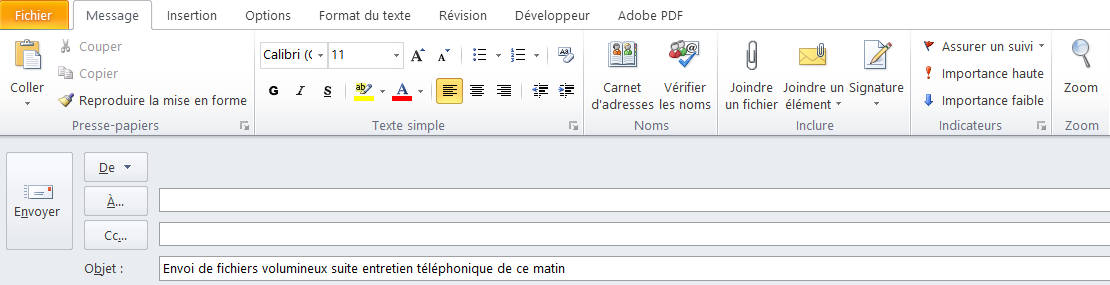
Pour joindre un ou des fichiers volumineux à un message en contournant les contraintes que nous venons de voir dans l’introduction, j’ai opté pour cette solution qui conserve une ergonomie très proche de celle utilisée ordinairement par les utilisateurs :

* l’utilisateur rédige son message et ajoute des pièces jointes comme d’ordinaire ;
* si un fichier dépasse le quota admis ou génère un message trop volumineux, le fichier n’est pas attaché mais juste une « image » de quelques octets, reprenant le nom du fichier précédé du repaire « ~~ » (deux tildes) ;
* puis il clique sur « Envoyer » comme à son habitude : les fichiers hors quotas sont automatiquement compressés en plusieurs lots de la taille admise, et si besoin plusieurs messages sont générés pour respecter la limite imposée.

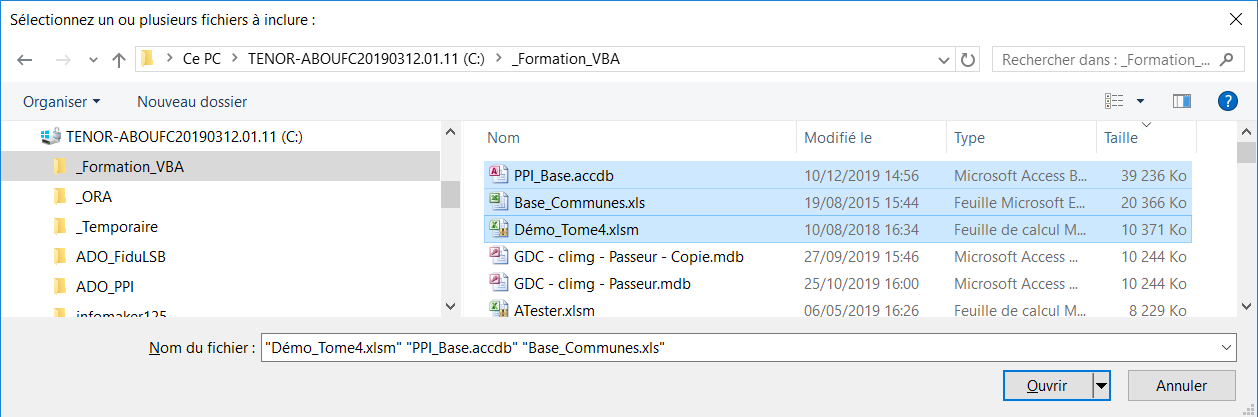
Exemple :

Pendant la rédaction d’un message, le bouton « Joindre un fichier » du ruban permet de sélectionner un ou des fichiers à joindre au message. Ne vous fiez pas aux apparences, c’est en fait un bouton personnalisé qui lance une procédure en VBA de sélection de fichiers. Ceci afin d’intercepter les fichiers sélectionnés et vérifier leur taille avant leur attachement, ce que le bouton standard ne permet pas de faire.

Le bouton standard est masqué pour éviter son utilisation, nous y reviendrons plus loin.



Ce qui ouvre un explorateur pour sélectionner les fichiers à joindre :

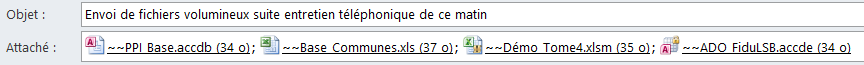


Cette opération est exécutée autant de fois qu’il le faut pour sélectionner les fichiers désirés, situés parfois dans différents répertoires.

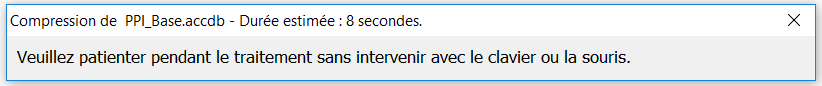
S’ils sont trop volumineux, les fichiers attachés ne sont pas les fichiers sélectionnés mais juste une « image » de quelques octets, reprenant le nom du fichier précédé du repaire « ~~ ».

Sinon, ils sont attachés comme d’habitude, c’est donc transparent pour l’utilisateur.

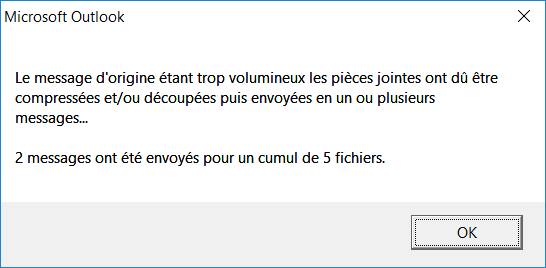
Ces pièces jointes peuvent être supprimées en cas d’erreur, comme pour les pièces jointes classiques.



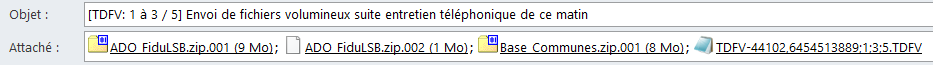
Quand l’utilisateur clique sur « Envoyer » les fichiers volumineux sont compressés et découpés en plusieurs lots, une boîte informe l’utilisateur sur l’avancement du traitement :



Une boîte annonce la fin de la génération des différents messages :



Les fichiers envoyés sont enregistrés dans le dossier « éléments envoyés », comme pour tout autre message ordinaire (si Outlook est configuré de la sorte) :



Notez que chaque objet du message est précédé du libellé « [TDFV: *x* à *y* / *z*] » où *x* et *y* indiquent les numéros des fichiers dans le message et *z* le nombre total de fichiers à transmettre, ceci afin que le destinataire s’assure avoir reçu tous les fichiers.

Ici « 1 à 3 / 5 » signifie que le message contient les fichiers 1, 2 et 3 sur les 5 fichiers à recevoir.

Les fichiers volumineux étant compressés en plusieurs lots, il y a ici 5 fichiers à transférer pour reconstituer les 4 fichiers d’origine.

Une pièce jointe est ajoutée à chaque envoi. Elle est d’extension « .TDFV » et contient la référence unique de traitement qui permettra par la suite d’identifier un transfert de fichiers volumineux.

Vous remarquerez que les fichiers compressés avec 7-Zip en plusieurs lots portent en extension « .zip.001 » à « .zip.*nnn* », nous y reviendrons plus tard dans cette documentation.

Et si vous ne disposez pas de 7-Zip ?

La procédure est identique. Les fichiers volumineux sont compressés avec Windows (l’équivalent du clic droit qui ouvre le menu contextuel « *Envoyer vers / Dossier compressé* », voir [l’annexe 1](#_Annexe_1_–)) puis, si besoin, sont découpés en plusieurs lots reconnaissables par leur extension « *.P000* » à « *.Pnnn* » (voir [l’annexe 2](#_Annexe_2_–)).

# Recevoir des fichiers volumineux avec Outlook

Nous venons d’étudier l’envoi des fichiers volumineux, parfois en plusieurs messages.

Voyons maintenant comment ça se passe du côté du destinataire.

En fait, rien de spécial non plus ici. Le destinataire détache les pièces jointes dans le dossier de son choix (sauf celle d’extension « .TDFV ») comme d’ordinaire, en s’assurant qu’il a bien reçu l’intégralité des fichiers attendus. C’est facile car l’objet des messages contient le libellé « [TDFV: *x* à *y* / *z*] » où *x* et *y* indiquent les numéros des fichiers dans le message et *z* le nombre total de fichiers à recevoir.

Et pour décompresser les fichiers compressés en lots avec 7-Zip d’extension « .zip.001 » à « .zip.*nnn* », pas de problème, il suffit de décompresser le fichier « .zip.001 » qui retrouvera automatiquement les autres « zip » qui lui sont associés.

L’utilisateur peut archiver ces messages dans le dossier de son choix, comme il le ferait avec des messages ordinaires.

Pour automatiser tout cela, j’ai développé trois fonctions que vous retrouverez dans le groupe « TDFV » du ruban lorsque vous ouvrez un message :



Le premier bouton « Récupérer les fichiers » permet d’extraire tous les fichiers d’un transfert de fichiers volumineux (ceux comportant le libellé clé « [TDFV:] » du message en cours de lecture) et de décompresser automatiquement ceux d’extension « .zip.*nnn* » (et recomposer ceux d’extension « *.Pnnn* »).

Ces fichiers sont enregistrés dans le répertoire de l’utilisateur « Transfert Des Fichiers Volumineux\Réceptions ».

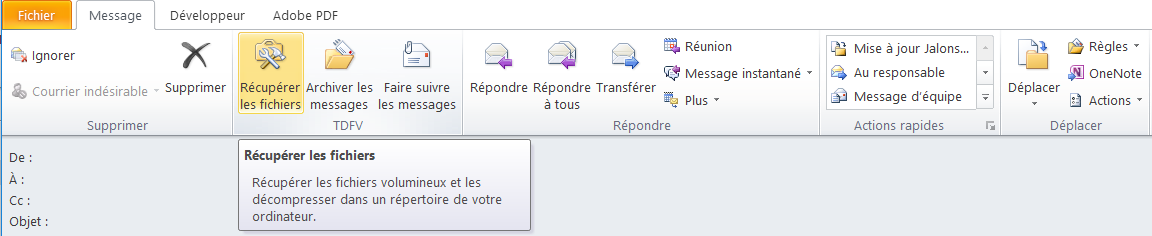
L’explorateur s’ouvre automatiquement à la fin du traitement, ce qui permet à l’utilisateur de les visualiser.

Le second bouton « Archiver les messages » déplace tous les fichiers d’un transfert de fichiers volumineux dans le dossier Outlook que choisi l’utilisateur.

Le troisième bouton « Faire suivre les messages » permet de faire suivre tous les fichiers d’un transfert de fichiers volumineux à d’autres destinataires. C’est l’équivalent de « Transférer » pour les messages classiques.

Exemple 1 :

L’utilisateur a reçu plusieurs messages suite à un transfert de fichiers volumineux. Il ouvre l’un de ces messages (pas forcement le premier) et clique sur le bouton « Récupérer les fichiers » :

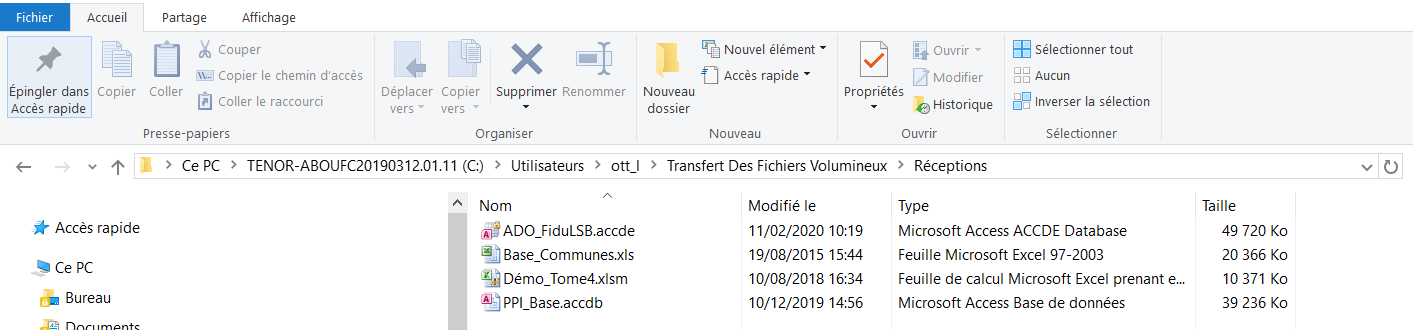


Outlook reconnaît automatiquement tous les messages du dossier actif contenant les fichiers du transfert, et décompresse les fichiers volumineux.

Une boîte informe l’utilisateur sur l’avancement du traitement :

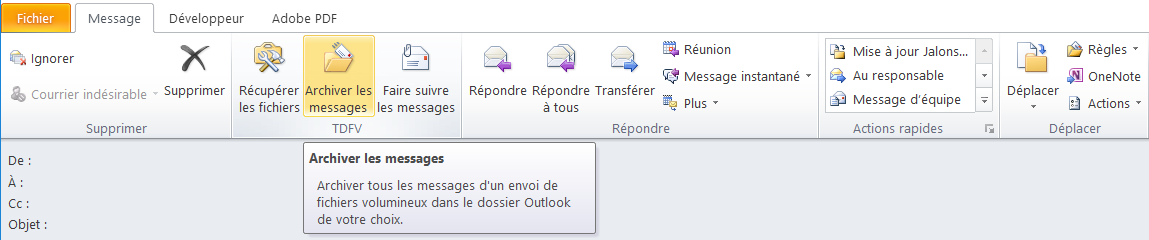


Puis ouvre l’explorateur sur le dossier de réception :

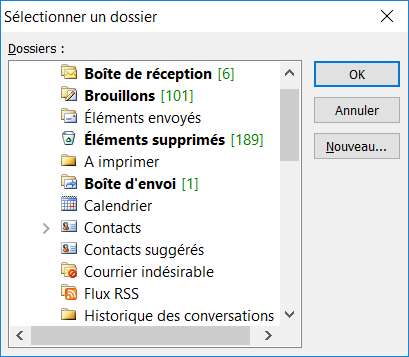


Exemple 2 :

Le bouton « Archiver les messages » ouvre la liste des dossiers disponibles :

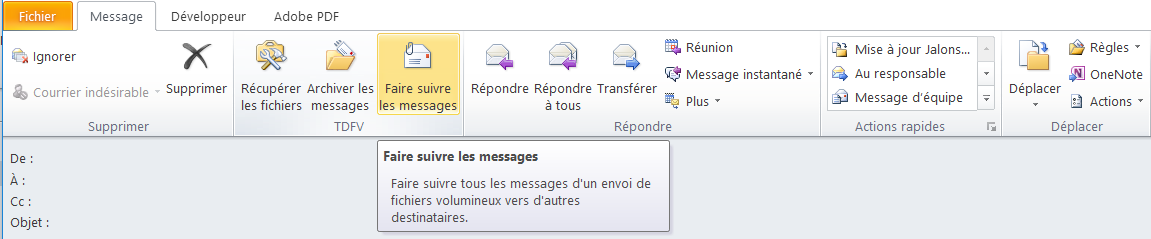


L’utilisateur sélectionne le dossier de son choix et clique sur « Ok » pour déplacer tous les messages du transfert en cours de lecture :



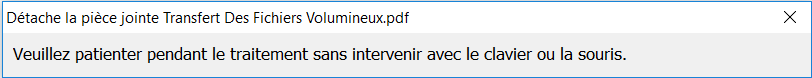
Exemple 3 :

L’utilisateur a reçu plusieurs messages suite à un transfert de fichiers volumineux. Il souhaite les faire suivre à un collègue qui en a également besoin. Il ouvre l’un de ces messages (pas forcement le premier) et clique sur le bouton « Faire suivre les messages » :



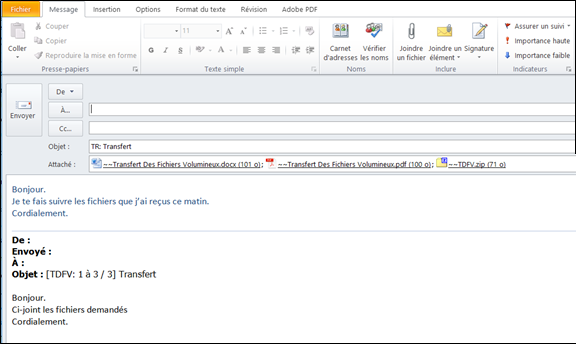
Outlook reconnaît automatiquement tous les messages du dossier actif contenant les fichiers du transfert, et détache leurs pièces jointes dans un répertoire temporaire.

Une boîte informe l’utilisateur sur l’avancement du traitement :



Puis un nouveau message est généré.

Ce message reprend le message d’origine et les images des pièces jointes. Comme pour un clic sur le bouton « Transférer » d’un message classique, l’utilisateur peut compléter le corps du texte, ajouter ou supprimer des pièces jointes, et clic sur « Envoyer » après avoir renseigné les destinataires.



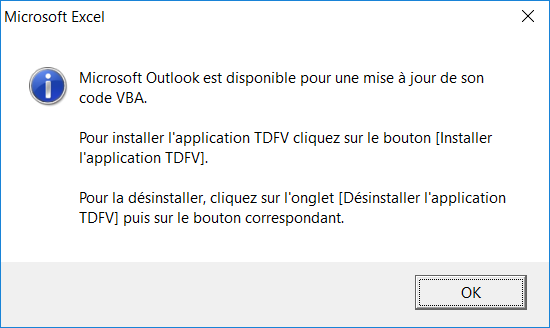
Un nouvel envoi de fichiers volumineux est déclenché, comme vu dans le premier exemple, en tenant compte des limitations de l’utilisateur (qui peuvent être différentes de celles de l’émetteur).

# Installer (et désinstaller) TDFV dans Outlook – Automatiquement avec Excel

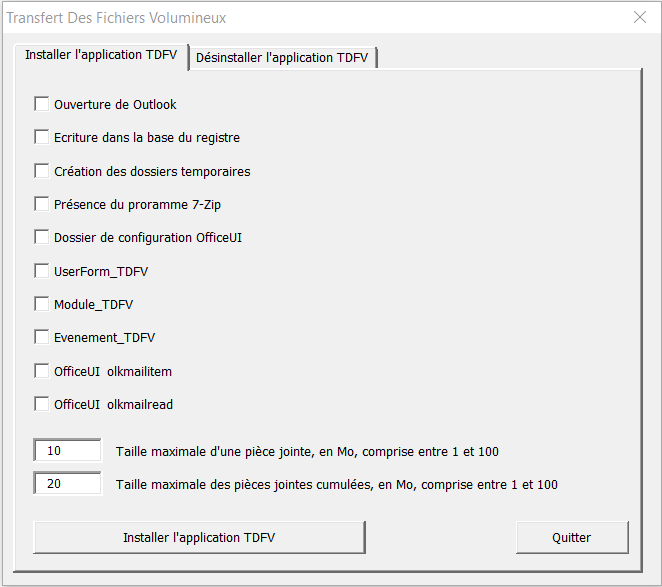
Si vous disposez d’Excel 2010 ou 2016 (version 32 bits) vous trouverez en annexe le fichier « Installation\_TDFV.xlsm » qui automatise l’installation des fonctionnalités et des boutons dans Outlook.

Pensez à activer les macros en lançant ce programme.

Après vérification de la présence d’Outlook (il est ouvert automatiquement s’il était fermé) et de sa disponibilité à être modifié, une boîte vous informe :



Avant de cliquer sur « Installer l’application TDFV », vous pouvez modifier les valeurs par défaut de la taille maximale d’une pièce jointe et de la taille maximale d’un envoi.



Par sécurité, veuillez à ne pas utiliser la souris et le clavier pendant l’installation des différents composants.

À mesure de l’avancement du traitement, les différentes étapes réalisées correctement se cochent.

Si tout se passe bien, un message vous informe qu’Outlook va être relancé.

Si vous avez reçu un message vous invitant à autoriser les macros, procédez ainsi : dans Outlook sur l’onglet *Fichier*, sélectionnez *Options*, *Centre de gestion de la confidentialité*, *Paramètres du Centre de gestion de la confidentialité*, *Paramètres des macros*, et cochez *Activer toutes les macros*.

Pour désinstaller automatiquement l’application :

Lancez ce programme et cliquez sur l’onglet « Désinstaller l’application TDFV » puis sur le bouton correspondant.

# Installer (et désinstaller) TDFV dans Outlook – Manuellement

Si l’installation automatique par Excel n’est pas possible, vous devrez installer manuellement les fichiers sources en VBA et les boutons dans Outlook.

Vous trouverez en annexe quatre fichiers sources : « Module\_TDFV.bas », « Evenements\_TDFV.cls », « UserForm\_TDFV.frm » et « UserForm\_TDFV.frx », repris dans le fichier « TDFV.zip » qu’il faut décompresser.

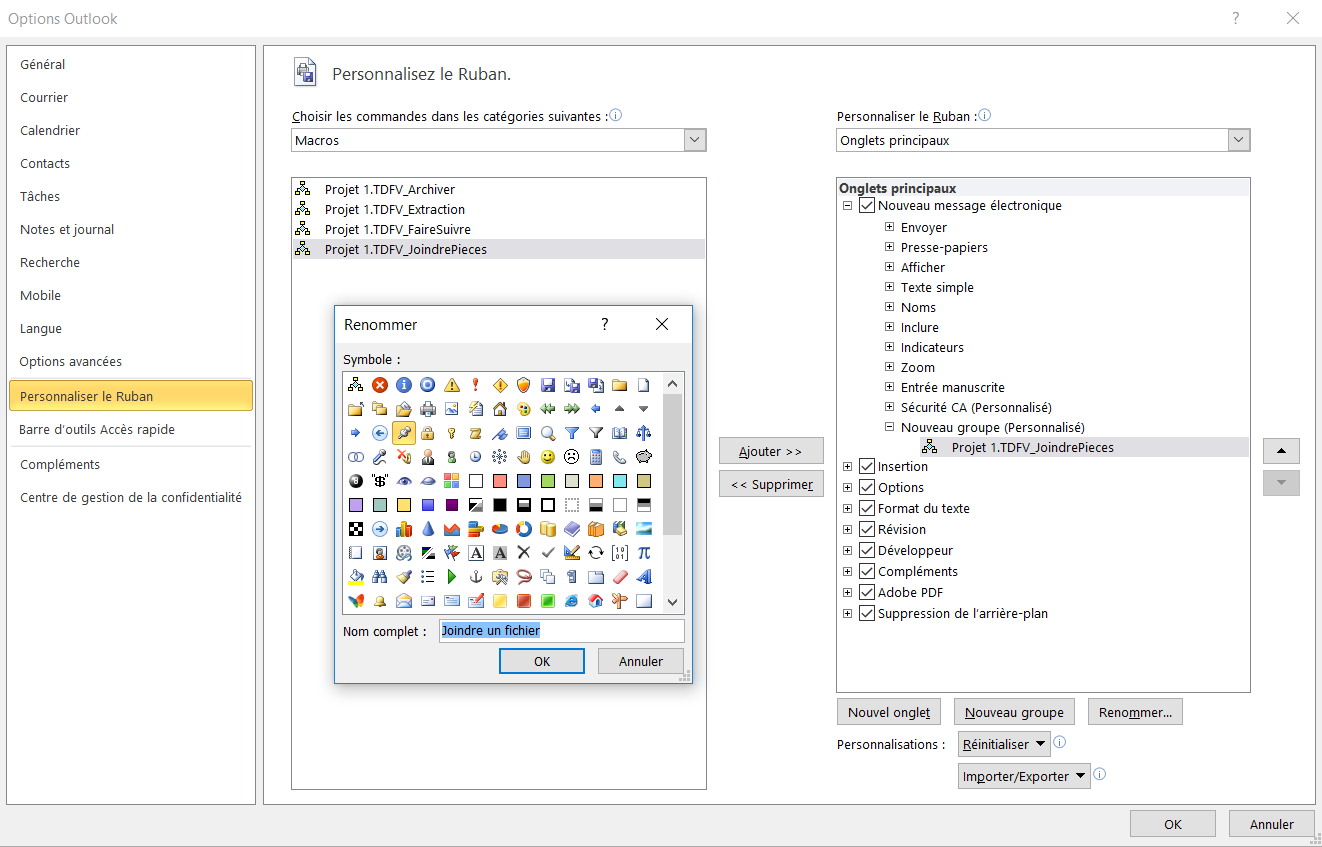
Voici comment les installer dans Outlook en quelques manipulations :

En premier lieu il faut ouvrir l’éditeur Visual Basic par la combinaison des touches [Alt]+[F11].

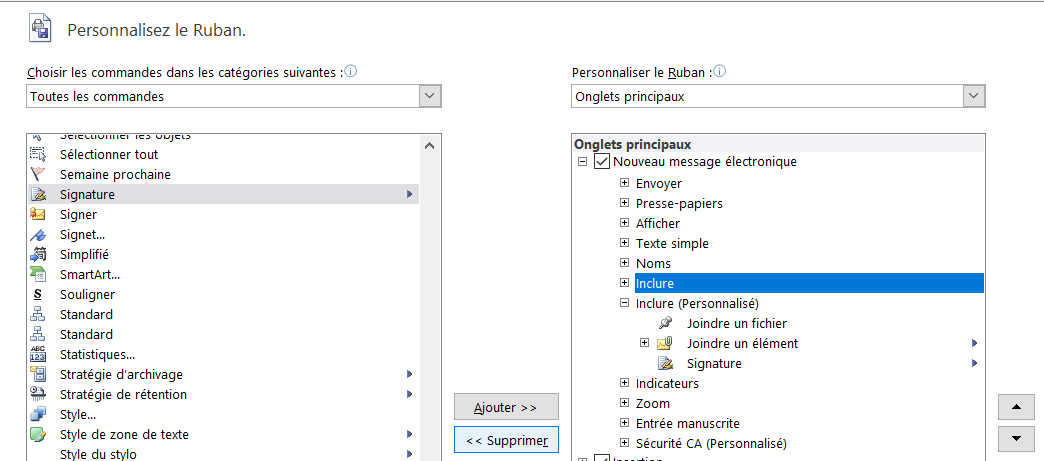
Puis importer les trois fichiers « Module\_TDFV.bas », « Evenements\_TDFV.cls », « UserForm\_TDFV.frm » par le menu « Fichier / Importer un fichier ». Le fichier « UserForm\_TDFV.frx » n’est pas à importer mais sa présence est indispensable pour configurer le formulaire « UserForm\_TDFV ».

Quittez l’éditeur et revenez sur l’écran d’accueil pour ajouter maintenant le bouton personnalisé « Joindre un fichier » et les commandes standards « Joindre un élément » et « Signature » au ruban des messages créés :

* cliquez sur « Nouveau message électronique » ;
* placez la souris au niveau du ruban et faites un clic droit et cliquez sur « Personnaliser le Ruban » ;
* dans la partie de droite, « Personnaliser le Ruban », activez « Nouveau message électronique » et cliquez sur « Nouveau groupe » ;
* dans la partie de gauche, « choisir les commandes dans les catégories suivantes », sélectionnez « Macros » et faites glisser/déposez la macro « Projet 1.TDFV\_JoindrePieces » sur le nouveau groupe qui s’est créé puis cliquez sur « Renommer » ;
* sélectionnez le symbole de votre choix dans la liste qui s’affiche (par exemple la puce) et modifiez le nom de la macro en « Joindre un fichier » puis validez sur « OK » ;

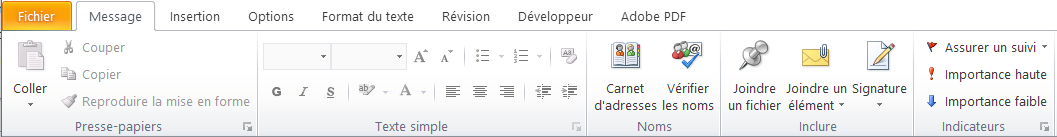


* sélectionnez le nouveau groupe et cliquez sur « Renommer » pour le renommer en « Inclure » (il n’est pas nécessaire de choisir un symbole car aucune icône ne sera affichée) ;
* dans la partie de gauche, « choisir les commandes dans les catégories suivantes », sélectionnez maintenant « Toutes les commandes » et faites glisser/déposez les commandes « Joindre un élément » et « Signature » sur le groupe « Inclure (personnalisé) » ;
* utilisez les flèches de la partie droite pour remonter le groupe créé sous le groupe « inclure » ;
* sélectionnez le groupe « inclure » et cliquez sur « Supprimer » ;



* validez vos modifications du ruban en cliquant sur « OK » en bas à droite.

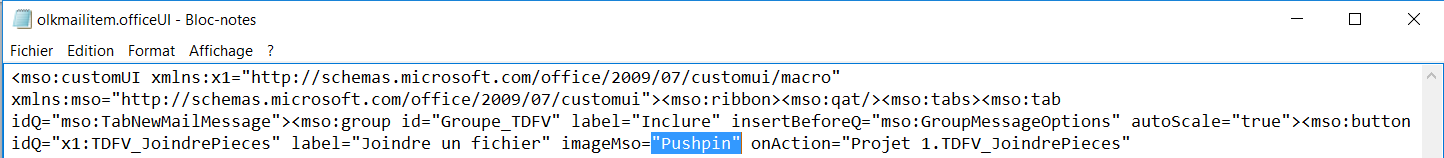
Le ruban est modifié et doit ressembler à cela :



Le choix des icônes proposées par Outlook est restreint, mais une astuce permet d’en changer en modifiant le fichier de configuration du ruban (au format XML). Ici c’est le fichier « olkmailitem.officeUI » qui se trouve :

* soit dans « C:\Users\USER\AppData\Local\Microsoft\Office » ;
* soit dans « C:\Users\USER\AppData\Roaming\Microsoft\Office ».

Ouvrez ce fichier avec un éditeur tel que le bloc-notes et repérez l’emplacement de l’icône (imageMso) utilisée (dans notre exemple c’est l’icône « Pushin »).



Remplacez le nom (« Pushin » dans notre exemple) par « AttachFile » et fermez le bloc-notes en enregistrant.

Vous trouverez la galerie des imagesMso à cette adresse : <https://codekabinett.com/download/Microsoft-Office-2010-imageMso-Gallery.pdf>

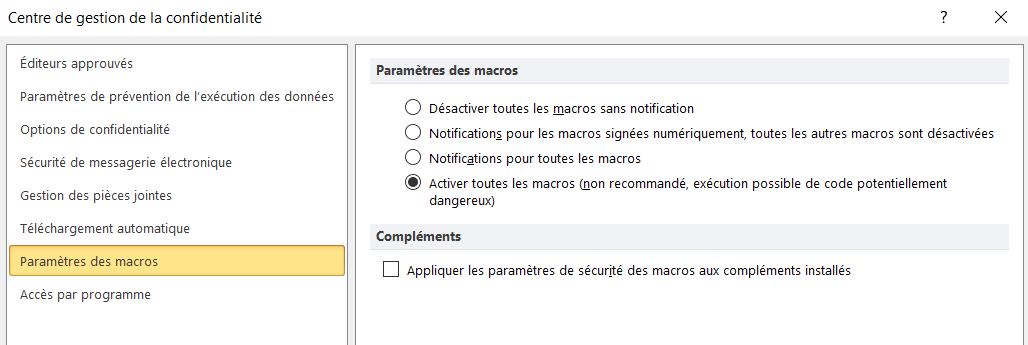
Après avoir ouvert un message pour accéder à son ruban, procédez de la même façon pour ajouter les boutons « Récupérer les fichiers », « Archiver les messages » et « Faire suivre les messages » au ruban.

Les macros concernées sont respectivement : « Projet 1.TDFV\_Extraction », « Projet 1.TDFV\_Archiver » et « Projet 1.TDFV\_FaireSuivre », à glisser dans le groupe à créer « TDFV ».

Le fichier de configuration du ruban est « olkmailread.officeUI ».

Les imageMso que j’ai utilisées sont : « GroupControls » pour récupérer les fichiers, « CreateMailRule » pour les archiver, et « ForwardAsAttachment » pour les faire suivre.

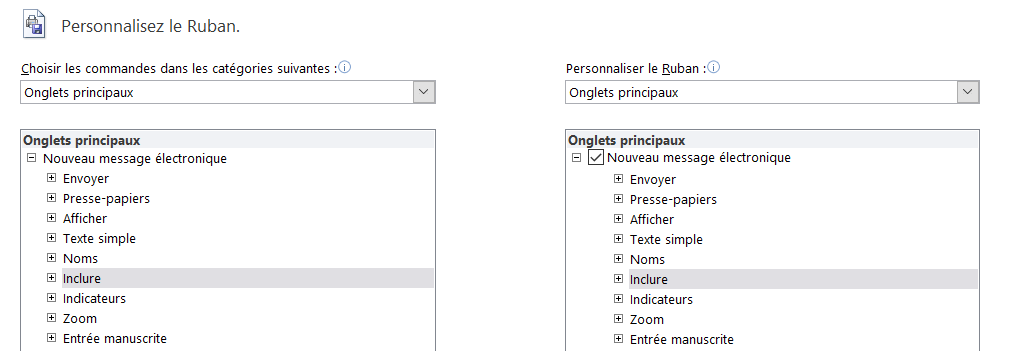
La dernière étape est d’activer les macros : dans Outlook sur l’onglet *Fichier*, sélectionnez *Options*, *Centre de gestion de la confidentialité*, *Paramètres du Centre de gestion de la confidentialité*, *Paramètres des macros*, et cochez *Activer toutes les macros*.



Quittez Outlook en confirmant l’enregistrement du projet VBA.

Pour désinstaller manuellement l’application :

* pour le code : ouvrir l’éditeur Visual Basic et pour chacun des éléments « UserForm\_TDF », « Module\_TDFV », « Evenements\_TDFV » faire un clic droit et choisir « Supprimer… » ;
* pour les boutons : ouvrir la personnalisation des deux rubans comme vu ci-dessus pour sélectionner les groupes créés et cliquer sur « Supprimer » ;
* pour réinstaller les commandes standards du groupe « Inclure » : dans la partie de gauche, « choisir les commandes dans les catégories suivantes », sélectionnez « Onglets principaux » puis « Nouveau message électronique » et sélectionnez « Inclure » et glisser/déposez sous le groupe « Noms » ;

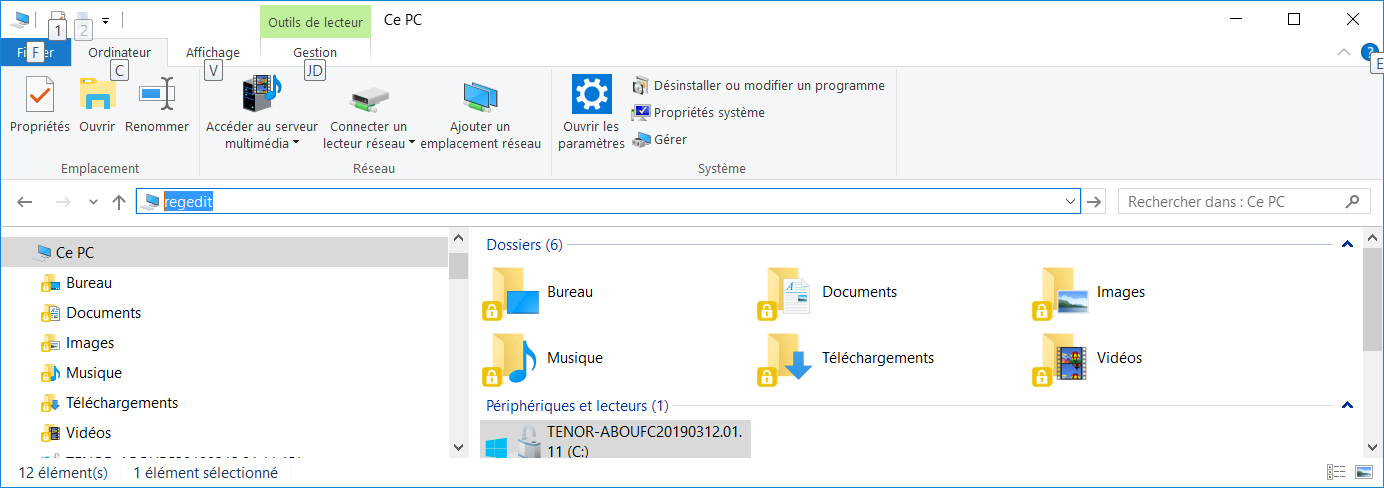


* validez vos modifications du ruban en cliquant sur « OK » en bas à droite.

Quittez Outlook sans oublier d’enregistrer le projet VBA.

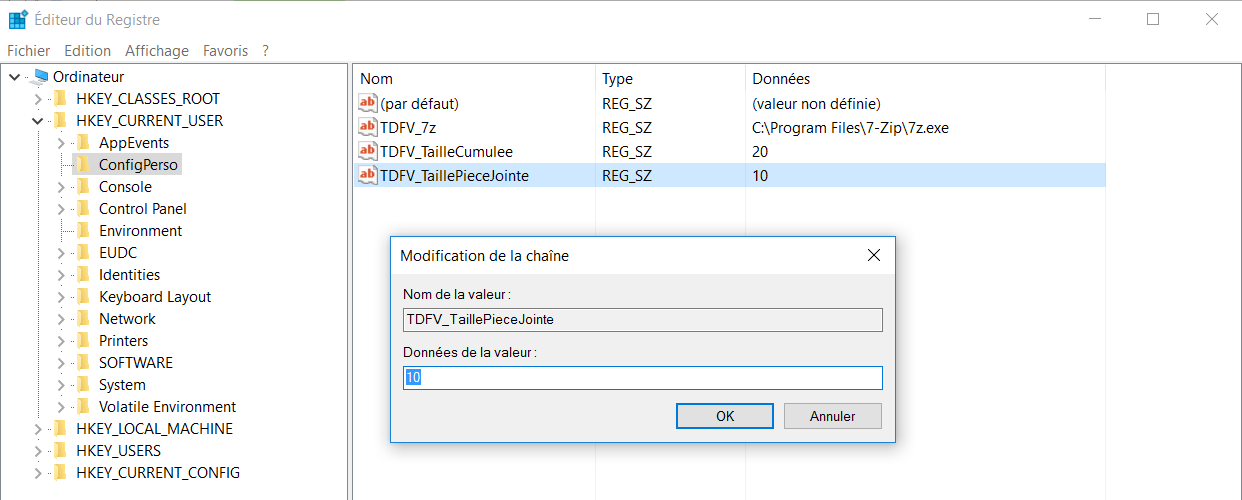
Pour modifier la configuration :

Par défaut, la taille maximale d’une pièce jointe est de 10 Mo et la taille d’un envoi de 20 Mo. Ces valeurs étant enregistrées dans le registre « ConfigPerso » de l’utilisateur (lors de la première utilisation des macros), elles peuvent être modifiées depuis l’explorateur en tapant « regedit » dans la barre d’adresse (mise ici en surbrillance bleu) :



Pour modifier la taille maximale d’une pièce jointe : double-cliquez sur « TDFV\_TaillePieceJointe ».

Pour modifier la taille maximale d’un envoi : double-cliquez sur « TDFV\_TailleCumulee ».



# Remarques sur le code VBA utilisé dans Outlook

Ces quelques remarques sur le code VBA utilisé dans Outlook, sans entrer dans le détail des 1 600 lignes du module « Module\_TDFV », vous permettront de mieux comprendre pourquoi j’ai retenu certaines solutions.

* Un clic sur « Joindre un fichier » lance la procédure « TDFV\_JoindrePieces ».

Cette procédure ouvre un explorateur pour sélectionner un ou plusieurs fichiers. Or il n’existe pas dans le VBA Outlook l’équivalent de la boîte de dialogue *FileDialog(msoFileDialogFilePicker)* bien connue dans le VBA Excel.

Une solution de contournement consiste à créer une cession Excel et l’utiliser pour appeler cette boîte de dialogue. Voir [ici](https://stackoverflow.com/questions/55754855/how-to-open-file-dialog-on-outlook-vba).

Seulement, j’ai constaté que parfois, et sans comprendre pourquoi, cet appel faisait planter Outlook. J’ai donc recherché une autre solution, et trouvé l’API *GetOpenFileName*. Voir [ici](https://jkp-ads.com/Articles/apideclarations.asp) pour la source d’inspiration et [ici](https://docs.microsoft.com/en-us/windows/win32/api/commdlg/ns-commdlg-openfilenamea) pour la configuration. Son usage est simplifié par la fonction *GetOpenFile* qui permet l’ajout d’éventuels filtres (option non utilisée dans cette application).

Pour chaque sélection faite, un contrôle de la taille des pièces jointes existantes et du fichier à joindre est réalisé :

For Each PJ In ActiveInspector.CurrentItem.Attachments

TaillePJ = TaillePJ + PJ.Size

Next PJ

Si les quotas sont respectés (avec une marge de sécurité de 1Mo incluse dans *TailleMaxi* et *TailleMaxiCumulee*) le fichier est attaché :

If FileLen(Fichiers(i)) < TailleMaxi \_

And FileLen(Fichiers(i)) + TaillePJ < TailleMaxiCumulee Then

ActiveInspector.CurrentItem.Attachments.Add Fichiers(i)

Sinon un fichier texte est créé dans le répertoire temporaire de l’application. Ce fichier ne contient que le chemin et le nom du fichier sélectionné :

Set f = CreateObject("Scripting.FileSystemObject").CreateTextFile(DossierTDFV & "~~" & StrNom & "." & StrExtension, True)

f.WriteLine (Fichiers(i))

f.Close

Et c’est ce fichier qui est ajouté aux pièces jointes du message actif :

ActiveInspector.CurrentItem.Attachments.Add DossierTDFV & "~~" & StrNom & "." & StrExtension

Par la suite, la lecture de cette image permettra de connaître les fichiers à compresser (chemin et nom) et à joindre au message.

Puis cette procédure initialise l’événement ItemSend : « *qui survient à chaque envoi d’éléments Microsoft Outlook, que ce soit par l’utilisateur … ou lorsque la méthode Send est utilisée dans un programme pour un élément Outlook, par exemple MailItem* ».

Set MyClassTDFV.MyOlAppTDFV = Outlook.Application

Contenu dans le module de classe « Evenements\_TDFV » :

'--------------------------------------------------------------------------------------------------

Private Sub MyOlAppTDFV\_ItemSend(ByVal Item As Object, Cancel As Boolean)

'--------------------------------------------------------------------------------------------------

Cancel = Main\_TDFV(Item)

End Sub

C’est ce qui permettra d’intercepter un clic de l’utilisateur sur le bouton « Envoyer » et lancera la procédure publique « Main\_TDFV » (du module Module\_TDFV) qui se chargera d’analyser les pièces jointes et de découper les fichiers volumineux si besoin. Elle retourne « Vrai » en cas d’erreur ce qui annule l’envoi (car *Cancel = True*).

Il est possible d’initialiser l’événement à l’ouverte d’Outlook, avec la procédure *Application\_Startup* placée dans *ThisOutlookSession*, mais cela nécessite d’intervenir dans cet objet qui peut contenir d’autres instructions, ce qui n’est pas sans risque.

À noter : à chaque message généré contenant des fichiers volumineux une référence de traitement est attribuée sous la forme d’une pièce jointe attachée au message d’extension « .TDFV ». Ceci pour retrouver par la suite tous les messages qui constituent le même envoi volumineux, et partagent donc cette référence.

Je me sers de cette méthode car Outlook ne propose pas une propriété « commentaire » disponible pour les programmeurs qui puisse être enregistrée et liée au message. Il existe bien la propriété *FlagRequest* mais l’inconvénient est qu’elle est effacée lorsque l’utilisateur met un suivi sur le message (les drapeaux du menu « assurer un suivi »).

* Utilisation de 7-Zip pour compresser un fichier en plusieurs fichiers.

La fonction *Shell* permet de lancer l’exécutable « 7z.exe » avec une ligne de commande qui contient les informations nécessaires mais elle a l’inconvénient d’être asynchrone, la suite du code VBA s’exécute alors que 7-Zip n’a pas terminé son traitement. Or, savoir quand la compression est terminée est indispensable pour envoyer les messages avec leurs pièces jointes correctement compressées.

L’astuce que j’ai retenue est d’utiliser les API [*OpenProcess*](http://allapi.mentalis.org/apilist/OpenProcess.shtml) et [*GetExitCodeProcess*](http://allapi.mentalis.org/apilist/GetExitCodeProcess.shtml) qui permettent de savoir quand un processus est terminé et donc de rendre *Shell* synchrone. Ce qui donne le code simplifié :

StrCommand = Chr(34) & ZipExe & Chr(34) & " a -tzip " \_

& Chr(34) & StrDossierDest & StrNom & ".zip" & Chr(34) & " " \_

& Chr(34) & StrFichier & Chr(34) \_

& " -v" & TailleMo & "m -sdel"

ShellAndWait(StrCommand, vbHide)

Où les variables représentent :

*ZipExe* = le chemin et nom de l’exécutable (par exemple « C:\Program Files\7-Zip\7z.exe »).

*StrDossierDest* = le dossier destination.

*StrNom* = le nom du fichier, sans l’extension .zip qui est ajoutée.

*StrFichier* = le chemin et le nom du fichier source.

*TailleMo* = la taille maximale d’un fichier en Mo.

Chr(34) correspond aux guillemets.

'-----------------------------------------------------------------------------------------------------

Public Function ShellAndWait(ByVal ShellStr As String, ByVal WindowState) As Long

'-----------------------------------------------------------------------------------------------------

Dim hProg As Long, hProcess As Long, ExitCode As Long

hProg = Shell(ShellStr, WindowState)

hProcess = OpenProcess(&H400, False, hProg) ' PROCESS\_QUERY\_INFORMATION

Do

GetExitCodeProcess hProcess, ExitCode

DoEvents

Loop While ExitCode = &H103 'STILL\_ACTIVE

End Function

* Un clic sur « Récupérer les fichiers » lance la procédure « TDFV\_Extraction ».

La première opération consiste à identifier la référence de traitement du message actif. Puis à boucler sur les messages de la boîte active à la recherche des messages contenant une référence de traitement compatible.

Vient ensuite le traitement des pièces jointes de chacun des messages pour les décompresser avec 7-Zip (ou les recomposer s’ils ont été découpés) dans un fichier temporaire sur le disque dur.

Les fichiers « .zip » sont supprimés, les fichiers définitifs sont déplacés dans le répertoire « Réceptions », puis l’explorateur est ouvert sur ce répertoire, sur le principe du répertoire « Téléchargements ».

L’utilisateur peut consulter les fichiers, les déplacer, les supprimer.

Vous remarquerez que si un utilisateur réceptionne des fichiers compressés par 7-Zip d’extension « .zip.*nnn* », il peut toujours les décompresser manuellement avec un logiciel compatible, mais pour les fichiers découpés (ceux d’extension « *.Pnnn* »), il lui faut impérativement l’application TDFV pour les reconstituer.

* Utilisation de 7-Zip pour décompresser un fichier.

Ici aussi *ShellAndWait* est utilisée pour lancer l’exécutable « 7z.exe » de façon synchrone, dans une fenêtre active mais masquée :

'----------------------------------------------------------------------------------------

Private Function TDFV\_UnZip(StrSource As String, StrDest As String) As Long

'----------------------------------------------------------------------------------------

Dim StrCommand As String

StrCommand = Chr(34) & ZipExe & Chr(34) & " e " \_

& Chr(34) & StrSource & Chr(34) \_

& " -o" \_

& Chr(34) & StrDest & Chr(34) & ""

TDFV\_UnZip = ShellAndWait(StrCommand, vbHide)

End Function

Le fichier à décompresser ayant été préalablement détaché sur le disque dur, le traitement prend rarement plus de deux secondes. Puis l’explorateur est lancé :

Call Shell("Explorer.exe " & DossierReception, vbMaximizedFocus)

* Un clic sur « Archiver les messages » lance la procédure « TDFV\_Archiver ».

Là aussi, la première chose à faire est d‘identifier la référence de traitement du message actif. Puis à boucler sur les messages de la boîte active à la recherche des messages contenant une référence de traitement compatible.

La sélection d’un dossier de destination peut se faire, cette fois, à l’aide d’une fonction du VBA Outlook :

Dim OLF As Outlook.MAPIFolder

Set OLF = olApp.GetNamespace("MAPI").PickFolder

Dim myDestFolder As Outlook.Folder

Set myDestFolder = GetFolder(OLF.FolderPath) ' GetFolder : exemples dans la documentation Outlook.

Le dossier source est connu :

Dim MaBoite As Outlook.Folder

Set MaBoite = ActiveExplorer.CurrentFolder

Chaque message est déplacé (en commençant par la fin et remontant vers le début pour ne pas en oublier) :

For i = MaBoite.Items.Count To 1 Step -1

MaBoite.Items(i).Move myDestFolder

Next i

* Un clic sur « Faire suivre les messages » lance la procédure « TDFV\_FaireSuivre ».

Après avoir identifié la référence de traitement du message actif, un nouveau message de type *Forward* est généré :

Dim ObjMail As MailItem

Set ObjMail = ActiveInspector.CurrentItem.Forward

Il faut supprimer les pièces jointes qui ont été reprises automatiquement du message actif (en commençant par la fin et remontant vers le début pour ne pas en oublier) :

For i = ObjMail.Attachments.Count To 1 Step -1

ObjMail.Attachments(i).Delete

Next i

Le dossier actif est connu :

Dim MaBoite As Outlook.Folder

Set MaBoite = ActiveExplorer.CurrentFolder

Puis l’on boucle sur les éléments de ce dossier à la recherche des autres messages qui partagent la référence de traitement, et dans ce cas, détacher les pièces jointes dans un dossier temporaire et les attacher au nouveau message créé : soit intégralement, soit uniquement leur image s’ils sont volumineux.

' Boucle sur la boîte active pour détacher les messages:

For i = 1 To MaBoite.Items.Count

' Si le message contient le mot clé:

If Left(ReferenceTDFV(MaBoite.Items(i)), 21) = Left(StrRéférence, 21) Then

' Détache les pièces jointes:

For Each PJ In MaBoite.Items(i).Attachments

If Right(PJ.FileName, 5) <> ".TDFV" Then

PJ.SaveAsFile DossierFaireSuivre & PJ.FileName

DoEvents

' Décompose le fichier:

Call DétailFichier(DossierFaireSuivre & PJ.FileName, StrRépertoire, StrNom, StrExtension)

' Compte la taille des pièces jointes existantes:

TaillePJ = 0

For Each NewPJ In ObjMail.Attachments

TaillePJ = TaillePJ + NewPJ.Size

If Left(NewPJ.FileName, 2) = "~~" Then TaillePJ = TailleMaxiCumulee + 1

Next NewPJ

' Si taille du fichier + taille pièces existantes le permet alors attache:

If FileLen(DossierFaireSuivre & PJ.FileName) < TailleMaxi \_

And FileLen(DossierFaireSuivre & PJ.FileName) + TaillePJ < TailleMaxiCumulee Then

ObjMail.Attachments.Add DossierFaireSuivre & PJ.FileName

' Sinon attache l'image et pas le fichier:

Else

' Fait une image:

Set f = CreateObject("Scripting.FileSystemObject").CreateTextFile(DossierTDFV & "~~" & StrNom & "." & StrExtension, True)

f.WriteLine (DossierFaireSuivre & PJ.FileName)

f.Close

Set f = Nothing

' Et attache cette image (au lieu du fichier):

ObjMail.Attachments.Add DossierTDFV & "~~" & StrNom & "." & StrExtension

End If

End If

Next PJ

End If

Next i

'--------------------------------------------------------------------------------

Private Function ReferenceTDFV(ByVal Item As Object) As String

'--------------------------------------------------------------------------------

Dim i As Long

If Item.Class <> Olmail Then Exit Function

' Analyse les piéces jointes du message et retourne la référence TDFV:

For i = 1 To Item.Attachments.Count

If Right(Item.Attachments(i).fileName, 5) = ".TDFV" Then

ReferenceTDFV = Left(Item.Attachments(i).fileName, Len(Item.Attachments(i).fileName) - 5)

Exit For

End If

Next i

End Function

# Remarques sur le code VBA utilisé pour l’installation automatique avec Excel

S’il est possible depuis certains langages de programmation de créer une cession Outlook pour naviguer dans ses dossiers, ses messages et autres éléments, le VBA Outlook n’a pas d’instruction pour importer un fichier ou personnaliser le ruban. Créer une cession Outlook n’est alors d’aucun recours.

La seule façon d’installer l’application TDFV est de simuler les mêmes interventions clavier que celles réalisées manuellement. Par exemple importer un fichier correspond à la combinaison des touches [Alt]+[F11] pour ouvrir l’éditeur Visual Basic, [Ctrl]+[m] pour ouvrir la boîte de dialogue d’importation, renseigner le chemin et nom du fichier, tabuler deux fois et envoyer [Entrée].

Pour simuler ces interventions j’ai utilisé Excel mais il est possible de réaliser la même chose avec d’autres langages de programmation.

Avec Excel l’envoi de touches peut se faire avec l’instruction *SendKeys* du VBA mais j’ai préféré utiliser les API :

* Pour trouver le *handle* (numéro attribué par le système à une application) de la fenêtre concernée et donner le focus à cette fenêtre avant de lui simuler une intervention clavier.
* Pour envoyer une combinaison de touches j’utilise l’API *keybd\_event* qui reconnaît les constantes des touches *vbKeys* : [Alt] = *vbKeyMenu*; [F11] = *vbKeyF11*; [Ctrl] = *vbKeyControl*; [m] = *vbKeyM*.
* Pour envoyer une chaîne j’ai repris la solution utilisée pour [CryptoVBA](http://laurent-ott.developpez.com/tutoriels/programmation-excel-vba-tome-4/) : copier la chaîne dans le presse-papiers, puis la coller avec [Ctrl]+[v].

Vous retrouverez toutes ces API, et les fonctions développées pour les utiliser facilement, dans le module « Module\_API ».

Et bien d’autres fonctions. Exemples…

La fonction *Hdc\_EnumérationHandle* alimente la variable *EnumChildListe()* de la liste des noms des boutons d’une fenêtre et retourne le nombre de boutons.

Je m’en sers après avoir quitté Outlook pour m’assurer que la fenêtre de confirmation d’enregistrement du projet, qui contient les trois boutons « Oui, Non, Annuler » s’ouvre, preuve que le traitement s’est déroulé correctement. Ne reste alors qu’à « cliquer » le premier bouton avec la fonction *Hdc\_BoutonCliquer* pour enregistrer les modifications.

La fonction *RegScanKey* scanne les clés du registre de l’utilisateur pour retrouver l’emplacement de la clé *Outlook\Security\Level* où est enregistré le niveau de sécurité d’accès aux macros (cet emplacement change suivant les versions d’Outlook). La fonction *OutlookSecurity* modifie cette clé à 1 pour forcer l’autorisation.

La mise à jour du ruban se fait en modifiant les fichiers de configuration *olkmailitem* et *olkmailread*.

Si le fichier n’existe pas, il faut le créer (au format XML) ; s’il existe, plusieurs situations sont possibles :

* il n’y a pas encore le groupe TDFV : on recherche la position de la balise *TabNewMailMessage* ou *TabReadMessage* pour savoir où ajouter le code du nouveau groupe ;
* le groupe existe déjà (cas d’une mise à jour) : on remplace la partie *Visible = Faux* par *Visible = Vrai*.

Pour désinstaller le groupe, il n’est pas nécessaire de supprimer le code, il suffit de remplacer la partie *Visible = Vrai* par *Visible = Faux*.

Le groupe « Inclure » standard se nomme *GroupIncludeMainTab*.

Toutes les fonctions sont dans le module « Module\_Outlook ».

Le dernier module de l’application, « Module\_Fichiers », contient les fonctions pour :

* copier, en binaire dans une feuille de calcul, les fichiers sources à importer dans Outlook ;
* reconstituer les fichiers sources à partir des données binaires stockées dans la feuille.

Cette méthode permet de n’avoir qu’un fichier à télécharger pour l’installation automatique.

# Conclusion

Nous venons de voir dans cette documentation qu’il est possible avec Outlook d’envoyer des fichiers volumineux (c’est-à-dire hors des quotas admis par votre fournisseur de messagerie) en les découpant en plusieurs fichiers et si besoin en les envoyant en plusieurs messages ; soit manuellement pour un usage ponctuel, soit de façon automatisée à l’aide du VBA de l’application TDFV.

Avant d’installer l’application TDFV, je vous invite à contacter votre administrateur pour vous assurer que vous êtes en conformité avec les règles de sécurité et les consignes de votre entreprise si vous utilisez Outlook pour le transfert de fichiers volumineux au lieu d’un service de partage.

À noter : la taille des quotas dépend de votre fournisseur de messagerie et peut varier des exemples présentés ici, qui sont les valeurs minimales des anciens standards.

Celles et ceux qui vont s’intéresser au code VBA utilisé par l’application TDFV et qui ne sont pas habitués à programmer sous Outlook trouveront une foule d’astuces dans la « FAQ Outlook » de developpez.com : <https://outlook.developpez.com/faq/>

Je n’ai pas souhaité ici détailler davantage le code de l’application car la FAQ très fournie, et qui m’a permis de réaliser une grande partie du code, répondra à toutes vos interrogations.

Quant aux débutants en VBA, je les invite à consulter le [tome 1](http://laurent-ott.developpez.com/tutoriels/programmation-excel-vba-tome-1/) de mes mémentos pour acquérir les bases de la programmation.

Attention, vous devez disposer de 7-Zip (version 4.14 de 2005 ou supérieure, voir [ici](https://www.7-zip.org/history.txt)) pour profiter de l’option qui permet de compresser un fichier en plusieurs lots, ce qui est utilisé par l’application pour optimiser l’envoi des pièces jointes volumineuses.

En cas de besoin vous pouvez télécharger une version récente sur ce site : <https://www.7-zip.fr/>.

Les tests ont été réalisés sous Windows 10 avec Outlook 2010 et 2016 et Excel 2010 et 2016 versions 32 bits.

Laurent OTT.

2020.

# Fichiers Joints

Les codes sources des modules à importer dans Outlook pour une installation manuelle de l’application TDFV, contenus dans le fichier « TDFV.zip » :

* Module\_TDFV.bas
* Evenements\_TDFV.cls

- UserForm\_TDFV.frm

- UserForm\_TDFV.frx

Un fichier pour Excel 32 bits qui contient les macros pour une installation (et une désinstallation) automatique de l’application TDFV :

* Installation\_TDFV.xlsm

# Annexe 1 – Compresser et décompresser un fichier avec Windows

Il existe une autre méthode en VBA pour compresser et décompresser un fichier au format « zip ».

Elle est utilisée dans cette application quand 7-Zip n’est pas présent. Ici aussi le traitement est asynchrone. Donc pour s’assurer que la compression est terminée, il faut boucler dans la cible en testant la présence du fichier source, avec une durée maximale (par exemple de 30 secondes) pour sortir de la boucle au cas où l’utilisateur clique sur le bouton « Annuler » de la boîte de dialogue qui s’ouvre lors de la compression des gros fichiers (qu’il est impossible de ne pas afficher en VBA).

Le principe pour compresser : un fichier est créé, ne contenant que les 23 caractères d’en-tête d’un zip. La source à compresser est copiée dans ce fichier avec [*CopyHere*](https://docs.microsoft.com/en-us/windows/win32/shell/folder-copyhere). Windows reconnaît le format zip et fait l’équivalent du clic droit qui ouvre le menu contextuel « *Envoyer vers / Dossier compressé* ».

Le principe pour décompresser : la méthode *CopyHere* est utilisée en indiquant la source et la destination.

Plus d’informations [ici](https://codekabinett.com/rdumps.php?Lang=2&targetDoc=create-zip-archive-vba-shell32).

Un code simplifié mais opérationnel :

'---------------------------------------------------------------------------------------

Public Sub ShellZip(ByVal ZipArchivePath As String, ByVal AddPath As String)

'---------------------------------------------------------------------------------------

' ZipArchivePath = Le fichier zip à créer (chemin + non).

' AddPath = Soit un fichier source (chemin + nom) soit un répertoire (sans le \ final).

' Exemple : call ShellZip("C:\\_Temporaire\Test.zip", "C:\ADO\_FiduLSB\ADO\_FiduLSB\_Bases\_3.accdb")

' Remarque : Si le fichier est trop long à compresser, une boîte de message système s'affiche.

'---------------------------------------------------------------------------------------

Dim ShellApp As Object, fSource As Object, fTarget As Object, iSource As Object, SourceItem As Object

Dim i As Long, lTimer As Variant, ContainingFolder As String, ItemToZip As String

If Dir(ZipArchivePath) <> "" Then Kill ZipArchivePath

Set ShellApp = CreateObject("Shell.Application")

CreateZipFile ZipArchivePath

Set fTarget = ShellApp.Namespace((ZipArchivePath))

ContainingFolder = Left(AddPath, InStrRev(AddPath, "\"))

ItemToZip = Mid(AddPath, InStrRev(AddPath, "\") + 1)

Set fSource = ShellApp.Namespace((ContainingFolder))

For i = 0 To fSource.Items.Count - 1

If fSource.Items.Item((i)).Name = ItemToZip Then

Set SourceItem = fSource.Items.Item((i))

Exit For

End If

Next i

fTarget.CopyHere SourceItem

lTimer = Timer

Do

DoEvents

Loop While fTarget.Items.Count <> 1 And lTimer + 30 > Timer

End Sub

'---------------------------------------------------------------------------------------

Private Sub CreateZipFile(ByVal fileName As String)

'---------------------------------------------------------------------------------------

Dim FileNo As Integer

Dim ZIPFileEOCD(0 To 22) As Byte

ZIPFileEOCD(0) = Val("&H50") ' 80

ZIPFileEOCD(1) = Val("&H4b") ' 75

ZIPFileEOCD(2) = Val("&H05") ' 5

ZIPFileEOCD(3) = Val("&H06") ' 6

FileNo = FreeFile

Open fileName For Binary Access Write As #FileNo

Put #FileNo, , ZIPFileEOCD

Close #FileNo

End Sub

'---------------------------------------------------------------------------------------

Public Sub ShellUnZip(ZippedFileFullName As Variant, UnzipToPath As Variant)

'---------------------------------------------------------------------------------------

' ZippedFileFullName = chemin + nom du fichier .zip

' UnzipToPath = répertoire de destination.

' Exemple : call UnzipAFile("C:\\_Temporaire\Test.zip", "C:\\_Temporaire")

'---------------------------------------------------------------------------------------

Dim ShellApp As Object

Set ShellApp = CreateObject("Shell.Application")

ShellApp.Namespace(UnzipToPath).CopyHere ShellApp.Namespace(ZippedFileFullName).Items

End Sub

# Annexe 2 – Découper un fichier en lots et le recomposer

La compression au format zip vu ci-dessus avec *ShellZip* ne permet pas de compresser un fichier en plusieurs lots, comme le fait 7-Zip. L’idée ici est de découper soi-même un fichier par lots. Les lots seront d’extension *Pnnn*, où *P000* est un fichier qui ne contient que le nom du fichier source, son extension, la taille des lots (en octets), la taille du fichier source, le nombre de lots générés.

Passez en arguments de la fonction *DecoupeFichier* : l’adresse du fichier à compresser ; la taille en Mo des lots.

Pour recomposer le fichier source, passez en argument de la fonction *RecomposeFichier* l’adresse du *P000*.

Un code simplifié mais opérationnel :

'---------------------------------------------------------------------------------------

Type RecordP000

Nom As String \* 255

Extension As String \* 16

TaillePaquet As Long

TailleTotale As Long

NbPaquets As Integer

End Type

'---------------------------------------------------------------------------------------

Public Sub DecoupeFichier(StrFichier As String, TailleMo As Long)

'---------------------------------------------------------------------------------------

Dim EnteteP000 As RecordP000

Dim FileNumberSource As Long, FileNumberDest As Long

Dim LenSource As Long, LenCopie As Long

Dim Paquet As Integer, i As Integer

Dim StrRépertoire As String, StrNom As String, StrExtension As String

Dim RecordUnK As String \* 1024

Dim TailleMaxi As Long

' Conversion en Mo:

TailleMaxi = TailleMo \* 1024 \* 1024

' Décompose StrFichier en répertoire, nom, extension:

StrRépertoire = Left(StrFichier, InStrRev(StrFichier, "\"))

StrNom = Mid(StrFichier, InStrRev(StrFichier, "\") + 1)

StrNom = Left(StrNom, InStrRev(StrNom, ".") - 1)

StrExtension = Mid(StrFichier, InStrRev(StrFichier, ".") + 1)

' Ouverture du fichier source par lot de 1 ko:

LenSource = FileLen(StrFichier)

FileNumberSource = FreeFile

Open StrFichier For Random As #FileNumberSource Len = Len(RecordUnK)

' Boucle sur le fichier en lisant 1024 octets à la fois (et pas un à un):

While LenCopie < LenSource

' Création du fichier (paquet) Destination si besoin,

' c'est-à-dire, 1er paquet ou taille maxi de l'ancien atteinte:

If LenCopie Mod TailleMaxi = 0 Then

If Paquet > 0 Then Close #FileNumberDest

Paquet = Paquet + 1

FileNumberDest = FreeFile

Open StrRépertoire & StrNom & ".P" & Format(Paquet, "000") For Random As #FileNumberDest Len = Len(RecordUnK)

End If

' Lecture du fichier source:

Get #FileNumberSource, , RecordUnK

' Ecriture dans le fichier destination:

Put #FileNumberDest, , RecordUnK

LenCopie = LenCopie + Len(RecordUnK)

Wend

' Fermeture des fichiers:

Close #FileNumberDest

Close #FileNumberSource

' Enregistrement de l'en-tête:

FileNumberSource = FreeFile

Open StrRépertoire & StrNom & ".P000" For Random As #FileNumberSource Len = Len(EnteteP000)

EnteteP000.Nom = StrNom & "?"

EnteteP000.Extension = StrExtension & "?"

EnteteP000.TaillePaquet = CLng(TailleMo)

EnteteP000.TailleTotale = CLng(LenSource)

EnteteP000.NbPaquets = CInt(Paquet)

Put #FileNumberSource, , EnteteP000

Close #FileNumberSource

End Sub

'---------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------

Public Sub RecomposeFichier(ByRef StrFichier As String)

'---------------------------------------------------------------------------------------

Dim EnteteP000 As RecordP000

Dim FileNumberSource As Long, FileNumberDest As Long, LenSource As Long, LenCopie As Long

Dim Paquet As Integer, i As Long, Taille As Long, TailleTotale As Long

Dim StrRépertoire As String, StrNom As String, StrExtension As String,

Dim RecordUnK As String \* 1024

' Lecture de l'en-tête:

FileNumberSource = FreeFile

Open StrFichier For Random As #FileNumberSource Len = Len(EnteteP000)

Get #FileNumberSource, , EnteteP000

StrNom = EnteteP000.Nom

StrExtension = EnteteP000.Extension

Taille = EnteteP000.TaillePaquet

TailleTotale = EnteteP000.TailleTotale

Paquet = EnteteP000.NbPaquets

Close #FileNumberSource

' Décompose StrFichier en nom, extension, répertoire:

StrNom = Left(StrNom, InStr(1, StrNom, "?") - 1)

StrExtension = Left(StrExtension, InStr(1, StrExtension, "?") - 1)

StrRépertoire = Left(StrFichier, InStrRev(StrFichier, "\"))

' Conversion en Mo:

Taille = Taille \* 1024 \* 1024

' Ouverture du fichier Destination d'extension provisoire ".~":

FileNumberDest = FreeFile

Open StrRépertoire & StrNom & ".~" For Random As #FileNumberDest Len = Len(RecordUnK)

Paquet = 0

' Boucles sur les différents paquets par lot de 1024 octets (1 ko):

While LenCopie < TailleTotale

' Ouverture du fichier Source, c'est-à-dire le premier paquet ou le suivant

' quand la taille maxi est atteinte:

If LenCopie Mod Taille = 0 Then

' Fermeture de l'ancien paquet:

If Paquet > 0 Then Close #FileNumberSource

' Ouverture du nouveau paquet:

Paquet = Paquet + 1

FileNumberSource = FreeFile

Open StrRépertoire & StrNom & ".P" & Format(Paquet, "000") For Random As #FileNumberSource Len = Len(RecordUnK)

End If

' Lecture de la source:

Get #FileNumberSource, , RecordUnK

' Cas particulier pour la fin:

If TailleTotale - LenCopie < Len(RecordUnK) Then

Close #FileNumberDest

FileNumberDest = FreeFile

Open StrRépertoire & StrNom & ".~" For Binary As #FileNumberDest Len = TailleTotale - LenCopie

Dim VarString As String

VarString = Left(RecordUnK, TailleTotale - LenCopie)

Put #FileNumberDest, LenCopie + 1, VarString

' Cas normal:

Else

Put #FileNumberDest, , RecordUnK

End If

' Compte le nombre d'octets copiés:

LenCopie = LenCopie + Len(RecordUnK)

Wend

' Fermeture des fichiers:

Close #FileNumberDest

Close #FileNumberSource

' Supprime le fichier d'origine:

Kill StrRépertoire & StrNom & "." & StrExtension

' Renomme le fichier temporaire en fichier d'origine:

Name StrRépertoire & StrNom & ".~" As StrRépertoire & StrNom & "." & StrExtension

' Supprime les paquets:

For i = 0 To Paquet

Kill StrRépertoire & StrNom & ".P" & Format(i, "000")

Next i

' Mémorise le nom définitif du ficher recomposé:

StrFichier = StrRépertoire & StrNom & "." & StrExtension

End Sub

'---------------------------------------------------------------------------------------

# Remerciements

Je remercie [Chrtoph](https://www.developpez.net/forums/u449691/chrtophe/) et [Pierre Fauconnier](https://www.developpez.net/forums/u27262/pierre-fauconnier/) pour l’intérêt qu'ils ont porté à cette documentation et leurs conseils.

Ainsi que X pour la correction orthographique.

Sans oublier celles et ceux qui voudront apporter un commentaire.