API Windows - Introduction

Catégorie : L'API Windows
Publié par myDearFriend! le 03-07-2006

VBA est un langage de développement très puissant et tellement riche qu'on en découvre tous les jours...c'est mon cas, pas vous ?

Toutefois, pour accroître d'avantage le potentiel à notre disposition et pallier aux limites qu'on rencontre parfois, il est possible d'avoir recours à l'utilisation de l'API Windows (Application P rogramming Interface) dans les procédures VBA "classiques". Vous accédez aux bibliothèques de fonctions et objets issues de DLL (Dynamic link library ou bibliothèque de liaisons dynamiques) du système d'exploitation et décuplez ainsi les possibilités de vos projets.

Tout d'abord, je tiens à préciser que je ne suis pas spécialiste sur le sujet, aussi, la quasi-totalité des exemples que vous trouverez dans cette section ont pour base des travaux réalisés par d'autres auteurs (que je cite s'ils sont connus) ou des procédures couramment utilisées. Avec une simple recherche sur le net, vous trouverez d'autres sources beaucoup plus pointues en la matière, n'hésitez pas à faire vos propres recherches et tester les exemples rencontrés. N'hésitez pas non plus à m'informer des éventuelles "coquilles" que vous pourriez découvrir en parcourant ces quelques pages... Même si j'ai mis au point et testé la totalité des exemples que je dépose ici, je ne suis pas à l'abri d'une erreur, d'une explication erronée ou incomplète.

Sachez toutefois que :

- L'utilisation de l'API spécifique Windows limite le fonctionnement de votre projet à ce système d'exploitation.
- Vous prenez le risque d'une dépendance de votre application aux versions des bibliothèques de fonctions auxquelles vous ferez appel.
- Plantages et autres bonnes surprises du genre sont le lot quotidien de ceux qui, comme moi non spécialiste, s'essayent au bricolage à base d'API Windows ! Prenez donc soin de toujours sauvegarder votre travail avant d'effectuer vos tests car les résultats sont parfois surprenant, pour ne pas dire déroutant ! Notez que la plupart du temps, les plantages ne pardonnent pas : au mieux, il conviendra de relancer Excel, au pire, il faudra relancer Windows !

Comment faire appel aux fonctions API Windows par le code VBA ?

En premier lieu, il convient de mentionner au projet VBA quelle est la fonction souhaitée (le nom de la fonction et ses arguments) et où la trouver (dans quelle bibliothèque ou DLL) ?

Pour répondre à ces 2 questions, on a recours à l'instruction Declare placée dans la zone de déclarations d'un module (c'est à dire dans son entête, avant toute procédure Sub ou Function). Le mot clé Declare va donc indiquer à VBA notre intention d'inclure dans le projet la référence à une fonction DLL externe.

La syntaxe utilisée est la suivante :

[Public|Private] **Declare** Sub|Function *Nom* Lib "*NomDeLaDLL*" [Alias "*NomAlias*"] [(*Liste arguments* As*types données*)]

1. Public|Private : mot clé (facultatif) utilisé selon la portée que l'on veut donner à la procédure (Public si la procédure doit être accessible depuis n'importe où dans le projet ou Private si on souhaite restreindre l'appel à l'unique module contenant la déclaration). Par défaut, la procédure est considérée comme Public.
2. Sub|Function : instruction qualifiant le type de procédure (Function si la routine doit retourner une valeur ou Sub si aucun retour n'est attendu).
3. Nom : Il s'agit ici du nom de la procédure. Ce sera le nom réel tel que défini dans la DLL, ou un nom libre si on a recours à l'Alias (voir point 5.). Attention! les noms sont sensibles à la casse. Le nom donné ici sera celui par lequel la procédure pourra être appelée dans le code VBA.
4. Lib "NomDeLaDLL" : La clause Lib indique quelle est la DLL visée, bibliothèque conteneur de la fonction souhaitée. S'il s'agit d'une librairie appartenant au sous-répertoire "System32" de Windows (ex: User32.dll, shell32.dll...), le nom seul de la DLL suffit (sans l'extension ".dll"). Si la DLL cible est située ailleurs, sans doute faudra-t-il préciser le chemin et le nom complet de la bibliothèque en question. D'une manière générale, si le chemin n'est pas spécifié, VBA va rechercher la DLL dans les répertoires suivants (dans l'ordre) : le répertoire courant, le sous-répertoire "...WindowsSystem32", le répertoire "...Windows" et dans les répertoires connus de la variable d'environnement Path.
5. Alias "NomAlias" : facultatif. Nom réel de la fonction dans la DLL. Ce mot clé est utilisé si le Nom donné à la procédure (voir point 3.) n'est pas le nom réel existant dans la bibliothèque. Il est parfois nécessaire de donner volontairement un autre Nom à la procédure, ceci afin d'éviter des conflits de noms déjà utilisés dans le code par exemple. Il peut également arriver que la syntaxe du Nom réel ne soit pas acceptée comme nom valide par VBA.
6. (Liste arguments As types données) : facultatif (dépend de la fonction ou procédure choisie).

Une fois déclarée, la procédure pourra donc être appelée dans le code VBA par son Nom et éventuels arguments attendus comme on le ferait pour n'importe quelle procédure "classique" VBA.

Quelles sont les fonctions API Windows disponibles et comment connaître les arguments attendus ?

C'est là toute la difficulté du sujet ! Même si on trouve de bonnes sources sur le Net notamment, il existe pourtant très peu de documentation claire (selon moi) en la matière. Les API sont malheureusement souvent documentées sur fond de langage C ou C++. Cela est normal puisque ces fonctions sont souvent programmées dans ces langages. Mais ça ne facilite évidemment pas le travail de ceux qui, comme moi, ne maîtrisent pas ce type de programmation.

Pour celles ou ceux qui désirent se lancer dans l'aventure et faire leur propre expérimentation, je recommande vivement de commencer par les liens ci-dessous qui représentent pour moi, le point de départ indispensable en la matière :

- MSDN Library : Procédure pas à pas - appel des API Windows
- AllAPI.net (avec son indispensable API-Guide, une base de données et exemples associés très détaillés)