# The REBOL Documentation Project

-- FR - Documentation REBOL - Divers --

## Divers

# Bye bye l'objet system

Philippe Le Goff

Première publication : 8 juin 2006, et mis en ligne le jeudi 8 juin 2006

**Résumé :**

Traduction de l'article 20 du blog REbol 3.0, sur l'objet system

Bye bye l'objet system ?

*Carl Sassenrath, CTO REBOL Technologies 4-May-2006 17:39 GMT Article #0020*

Bien... si toutes ces discussions sur REBOL 3.0 ne vous ont pas fait réfléchir, voici quelque chose qui le fera...

Je pense y avoir fait allusion avant : l'objet system est problématique car il induit implicitement un modèle de partage. (Je savais cela lorsque je l'ai introduit, et je savais qu'un jour viendrait où cela devra être changé.)

Le problème :

L'objet system permet d'accéder à la plupart des informations sur l'état interne de REBOL. Cette information est souvent renvoyée sous la forme d'objets. Au sein d'un système multitâche et modulaire, de tels objets sont problématiques.

Il y a deux problèmes majeurs :

1. La plupart de ces objets devront devenir des modules indépendants. Par exemple, l'environnement View devra devenir un module de REBOL, qui exportera les fonctions nécessaires à votre programme. (Ne vous inquiétez pas, vous pourrez encore obtenir le module View et son code, mais l'accès se fera d'une manière plus spécifique.)

2. Si vous avez à partager des objets entre modules et tâches, que se passera-t-il Si l'un des modules/tâches modifie l'objet partagé ? Est-ce que ce sera "surprise !" dans les autres modules. Cela peut aussi devenir un trou de sécurité.

La Solution :

Nous allons avoir besoin, soit de supprimer l'objet system, soit au moins de changer la manière dont l'objet system fonctionne et ce qu'il contient.

Regardons cela d'un peu plus prés. Si vous tapez : "help system", voici ce que vous obtenez :

version tuple ! 1.3.2.3.1 build date ! 5-Dec-2005/18:22:54-8:00 product word ! View core tuple ! 2.6.3 components block ! length : 54 words object ! [unset ! error ! datatype ! context ! native ! action ! ... license string ! *REBOL End User License Agreement IMPORTANT. READ ...* options object ! [home script path boot args do-arg link-url server... user object ! [name email home words] script object ! [title header parent path args words] console object ! [history keys prompt result escape busy tab-size b... ports object ! [input output echo system serial wait-list] network object ! [host host-address] schemes object ! [default Finger Whois Daytime SMTP ESMTP POP IMAP ... error object ! [throw note syntax script math access command resv... standard object ! [script port port-flags email face sound] view object ! [screen-face focal-face caret highlight-start high... stats native ! System statistics. Default is to return total memo... locale object ! [months days] user-license object ! [name email id message]

Vous pouvez séparer cette liste en différentes catégories de solution :

Bloqué Certain de ces champs ne sont pas un problème car nous pouvons bloquer leur modification. Par exemple, les programmes ne pourraient pas modifier cette genre de champ. Beaucoup d'autres tombent dans cette catégorie.

Local Certains champs sont "relatifs au contexte". Par exemple, un module dans votre script ("l'environnement global de votre script") pourrait accéder à l'objet system, mais les autres modules ne le pourraient pas. Certains champs pourraient être "relatifs au contexte".

Remplacé Certains champs comme schemes et View sont remplacés par des modules. Je ne suis pas sûr que ces champs aient encore du sens ici, dans l'objet system.

Supprimé Par exemple, l'objet words est supprimé. Il n'y a plus de liste globale de mots et de valeurs. Les mots sont relatifs à leurs modules. (Mais il peut être possible d'avoir une liste de mots pour contenir les mots et valeurs du module courant, donc peut-être qu'elle pourrait être déplacée dans la catégorie "locale" ci-dessus.

System en tant que fonction ?

Une solution pourrait être de transformer system en une fonction et plus en objet. Ceci pourrait être fait pour fonctionner avec de nombreux appels :

```
print system/version
```

L'avantage est que la fonction system pourrait dynamiquement créer les valeurs demandées.

Mais cette approche fonctionnelle va introduire une rupture dans les scripts qui définissent certaines valeurs directement, comme avec :

```
system/options/binary-base : 64
```

(Ceci étant, cela vous fait penser qu'une telle notation suggère une fonction set-path qui peut avoir son intérêt pour des fonctions. Mais, ne nous laissons pas distraire pour le moment).

Voilà, vous avez matière à réflexion. Le problème n'est pas urgent actuellement, mais il sera nécessaire de le résoudre dans les semaines à venir.

(Traduction : Philippe Le Goff)