UTILISATION DES INSTRUCTIONS FILENAME ZIP ET SFTP

SAS 9.4 permet d'accéder à des fichiers compressés via la nouvelle méthode d'accès ZIP dans l'instruction FILENAME. Cet article va décrire son utilisation au travers d'exemples concrets. Il va aussi présenter la méthode d'accès SFTP de l'instruction FILENAME existant depuis SAS 9.2 mais peu illustrée.

Caractéristiques :

Catégories : Base
OS : Windows, Unix, z/OS
Version : SAS® 9.4
Vérifié en septembre 2016

Sommaire

1. Méthode d'accès ZIP 1
1.1. Création ou modification d'une archive ZIP 2
1.2. Lecture de données depuis une archive ZIP 3
1.2.1. Lecture d'un fichier connu de l'archive zip 3
1.2.2. Lister les fichiers d'une archive afin de les exploiter 4
2. Méthode d'accès SFTP 6
2.1. Principe et prérequis 6
2.2. Exemple d'utilisation 6
2.2.1. Génération des clés et préparation 6
2.2.2. Utilisation 10
3. Liens utiles 12
4. Conclusion 12

1. METHODE D'ACCES ZIP

Cette méthode d'accès ZIP utilisable dans l'instruction FILENAME est une nouveauté de la version SAS 9.4. Elle permet, en mode dit « lecture », d'accéder directement à des fichiers compressés dans une archive zip (créée avec un logiciel de compression tel que 7-zip, Winzip ou gzip).
Elle permet également, en mode dit « écriture », de créer ou mettre à jour une archive zip en y ajoutant ou supprimant des fichiers.
Elle s'utilise dans l'instruction FILENAME et est documentée sur cette page :
Filename ZIP
Un point important est que cette méthode ne permet pas de lire des fichiers zip qui ont été protégés (archivés) avec un mot de passe. Elle ne permet pas non plus de créer de fichiers zip protégés par mot de passe.

Nous allons voir comment utiliser cette méthode d'accès sur des exemples pratiques.

1.1. Création ou modification d'une archive ZIP

Nous allons créer une archive ZIP en y ajoutant des données provenant de 2 tables de la bibliothèque SASHELP, soit :

- CARS dont nous ne prendrons que les colonnes Make, Model, Type et Origin,
- PRDSALE dont nous ne prendrons que les colonnes ACTUAL, COUNTRY, PRODUCT YEAR et MONTH.

Il est nécessaire de passer par des étapes DATA successives afin de créer chaque fichier via l'instruction FILE sinon une erreur sera renvoyée. Ces fichiers seront de type texte. On peut spécifier les formats des colonnes qui vont être écrites sinon SAS appliquera le format défini pour la colonne. Le code SAS suivant est commenté, notamment sur les options intéressantes telles que « COMPRESSION » qui permet d'indiquer le niveau de compression voulu et « DEBUG » pour avoir plus de trace dans le journal d'exécution, et implémente ce cas de figure.

```
/* Archive zip à créer */
%let ziparchive=c:\tests\testzip.zip;
/* On demande un niveau de compression de 8 sur une échelle de
0 : pas de compression
à
9 : compression maximale
Valeur défaut 6
On met aussi l'option debug pour avoir plus de trace dans la log
*/
filename archive ZIP "&ziparchive" compression=8 debug;

/* Premier fichier à archiver issu de la table SASHELP.CARS */
data _null_;
  /* Creation ou ajout du fichier */
  file archive(cars.txt);
  set sashelp.cars;
  put Make $13. Model $40. Type $8. Origin $6.;
run;

/* Second fichier à archiver issu de la table SASHELP.PRDSALE */
data _null_;
  /* Creation ou ajout du fichier */
  file archive(prdsale.txt);
  set sashelp.prdsale;
  /* On ne spécifie pas tous les formats des colonnes */
  put Actual Country $10. Product Year Month;
run;
```

Il est à noter que l'archive zip est créée si elle n'existe pas. Si elle existe, elle est mise à jour (par exemple si elle contient d'autres fichiers archivés, ceux-ci sont conservés). Cela signifie que si nous exécutons ensuite ce code SAS :

```
data _null_;
  file archive(prdsale2.txt);
  set sashelp.prdsale;
  put Actual Country $10. Product Year Month;
run;
```

L'archive zip résultante contiendra 3 fichiers : « cars.txt », « prdsale.txt » et « prdsale2.txt ».

Il est aussi possible de supprimer un fichier dans une archive zip en utilisant la fonction FDELETE. En poursuivant sur notre exemple, le code suivant met à jour l'archive zip précédemment créée et contenant 3 fichiers, en supprimant le dernier ajouté, soit « prdsale2.txt ».

```
/* Assignation du fichier à supprimer dans l'archive */
filename archive ZIP "&ziparchive" member="prdsale2.txt";
data _null_;
 /* Test que ce fichier existe bien dans l'archive et si oui suppression
*/
 if (fexist('archive')) then do;
  rc = fdelete('archive');
  /* Le code retour de l'opération doit être 0 */
  put "Return code =" rc;
 end;

run;
```

1.2. Lecture de données depuis une archive ZIP

Cette fonctionnalité offre plus de possibilités. Toujours sur des exemples concrets nous allons explorer celles-ci.

1.2.1. Lecture d'un fichier connu de l'archive zip

Nous allons exploiter l'archive zip créée au paragraphe précédent pour lire les données du fichier texte « cars.txt » afin de constituer une table SAS correspondante. Cela suppose que l'on connaît les noms et contenus précis de cette archive. Là encore une étape DATA va être utilisée.
Le code SAS suivant effectue ces actions et il est commenté afin d'expliquer « pas à pas » ce qui est exécuté :

```
/* Archive zip dont on doit extraire le fichier 'cars.txt' */
%let ziparchive=c:\tests\testzip.zip;
filename archive ZIP "&ziparchive";

/* Table WORK.CARS_UNZIPPED construite depuis le fichier extrait */
data work.cars_unzipped;
  /* Extraction du fichier 'cars.txt' de l'archive zip */
  infile archive(cars.txt);

  /* Description des colonnes qui vont être constituées depuis le fichier
*/
  input Make $13. Model $40. Type $8. Origin $6.;
run;

/* Affichage de la table résultantes créée */
title "Données de Cars_unzipped";
proc print data=work.cars_unzipped;
run;
```

Un autre moyen de spécifier le fichier à extraire est d'utiliser le paramètre MEMBER dans l'instruction FILENAME. Sur notre exemple cela donne ceci :

```
filename archive ZIP "&ziparchive" member="cars.txt";
```

Il faut ensuite dans l'étape DATA indiquer simplement :

```
infile archive;
```

1.2.2. Lister les fichiers d'une archive afin de les exploiter

Dans l'exemple précédent nous connaissions les noms des fichiers présents dans l'archive zip. Nous allons maintenant utiliser le code suivant qui permet d'analyser le contenu de l'archive zip afin d'en obtenir la liste des fichiers.

En continuant sur l'archive zip créée aux étapes précédentes voici le code commenté, toujours basé sur une étape DATA, qui permet d'avoir dans une table SAS la liste des fichiers :

```
/* Assignation de l'archive zip  */
%let ziparchive=c:\tests\testzip.zip;
filename archive ZIP "&ziparchive";

/* Constitution de la liste des fichiers dans la table MEMBERS */
data work.members(keep=fichier);
 length fichier $200;
 /* Ouverture de l'archive et arret si erreur constatée */
 fid=dopen("archive");
 if fid=0 then
  stop;
 /* Comptage du nombre de fichiers dans l'archive */
 filecount=dnum(fid);
 /* Boucle pour récupérer la liste des noms de fichiers */
 do i=1 to filecount;
  fichier=dread(fid,i);
  output;
 end;
 /* Fermeture de l'archive */
 rc=dclose(fid);
run;

/* Affichage de la liste  */
title "Fichiers dans l'archive zip";
proc print data=work.members ;
run;
```

Le résultat affiché :

Fichiers dans l'archive zip

Obs.	fichier
1	cars.txt
2	prdsale.txt

Bien sûr ces cas sont très simples, cependant ils décrivent les principes généraux à suivre.

Pour terminer avec le FILENAME ZIP et sur un exemple un peu plus élaboré, nous allons travailler avec un fichier zip qui contient entre autres un fichier Excel qui pourra ensuite être exploité (importé en table SAS par exemple via une PROC IMPORT). Le fichier zip contient un fichier Excel nommé « cars.xlsx » et le code commenté suivant va permettre de le décompresser dans le répertoire de la WORK SAS afin de l'exploiter ensuite :

```
/* Assignation du fichier zip */
filename inzip ZIP "c:\tests\myzip.zip";
/* Fichier Excel décompressé sera mis dans le répertoire de la WORK */
filename xl "%sysfunc(getoption(work))/cars.xlsx" ;

/* Décompression via une étape DATA */
data _null_;
   /* Définition des attributs spécifiques pour la décompression */
   infile inzip(CARS.xlsx)
       lrecl=256 recfm=F length=length eof=eof unbuf;
   file   xl lrecl=256 recfm=N;
   input;
   put _infile_ $varying256. length;
   return;
 eof:
   stop;
run;

/* Importer le fichier Excel décompressé dans SAS */
proc import datafile=xl ...;
```

Ce dernier exemple est tiré d'un article du blog de Chris Hemedinger « The SAS Dummy » qui traite de ce sujet et aborde des points tels que :

- Récupérer la liste des fichiers d'une archive zip car en général c'est structuré sous forme d'une arborescence de répertoires et fichiers (comme dans un « explorateur » Windows). En effet notre premier exemple utilisait une archive zip qui contenait des fichiers à un seul niveau et sans répertoire.
- Décompresser les fichiers de type Excel ou tables SAS afin de les exploiter.

Voici le lien vers cet article :
SAS Dummy FILENAME ZIP
Egalement un autre article en relation sur ce blog :
SAS Dummy FILENAME ZIP other

Pour terminer avec le sujet « zip », il est à noter que l'instruction ODS PACKAGE permet aussi de créer des fichiers zip depuis SAS 9.2. Cette SAS Note indique comment procéder :
http://support.sas.com/kb/45/277.html
Egalement cet article en relation :
http://support.sas.com/resources/papers/proceedings09/018-2009.pdf

2. METHODE D'ACCES SFTP

Cette méthode d'accès est disponible depuis SAS 9.2 et permet d'assigner une ressource sur un serveur SFTP afin de l'exploiter. Elle est peu documentée et nécessite des prérequis et une mise en œuvre préparatoire comme nous allons le voir.

2.1. Principe et prérequis

Cette méthode d'accès, quand elle est appelée, lance dynamiquement un client SFTP qui doit par conséquent être présent et installé. Ce client va initier la connexion sécurisée et cryptée avec le serveur SFTP. Le trafic réseau entre le client et le serveur sera donc protégé et indéchiffrable à la différence d'une session FTP ou il passerait « en clair ».

Toutefois avant de pouvoir utiliser la méthode SFTP dans SAS il est nécessaire de configurer à la fois le client et le serveur SFTP. De plus seules certaines implémentations sont **supportées**.
Les prérequis sont :

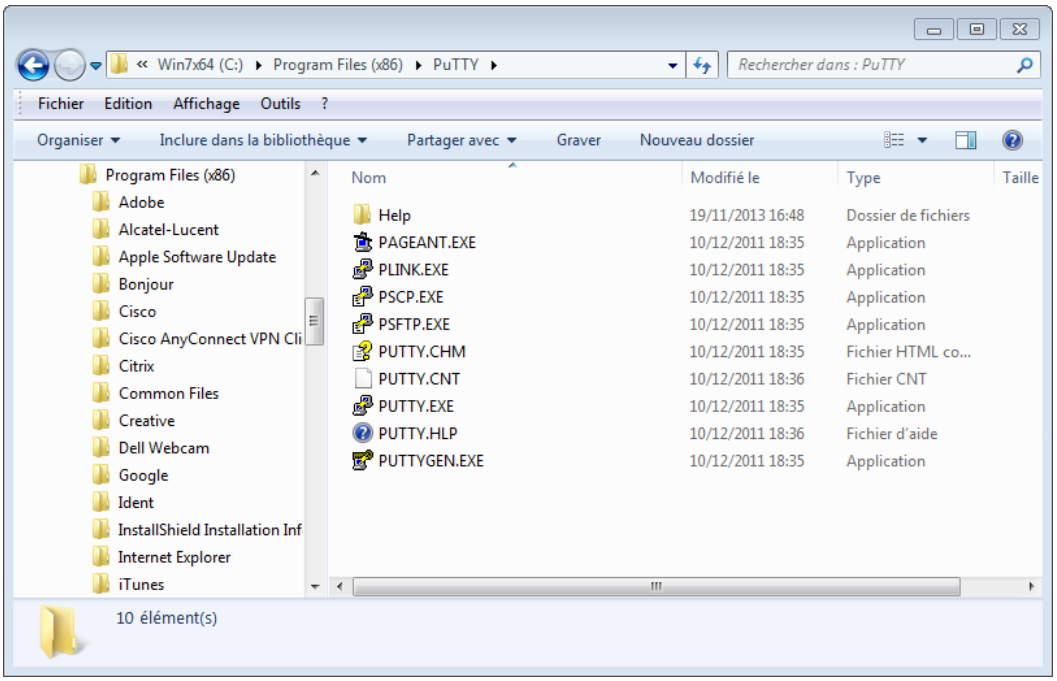
- Le serveur SFTP doit reposer sur OpenSSH (site : http://www.openssh.com), toutefois la plupart des serveurs Unix/Linux utilisent ce serveur donc ce n'est pas une contrainte,
- Le client SFTP doit utiliser le logiciel PuTTy si vous êtes sous Windows ou OpenSSH (client) si vous êtes sous Unix/Linux,
- Le protocole SSH en version 2 doit être utilisé (ce n'est pas une contrainte car c'est la norme, en effet le protocole SSH en version 1 est historique, peu sécurisé et de ce fait obsolète),
- Une authentification par couple clé privée/clé publique,
- Un « agent » côté client pour gérer la partie négociation clé privée/clé publique.

2.2. Exemple d'utilisation

2.2.1. Génération des clés et préparation

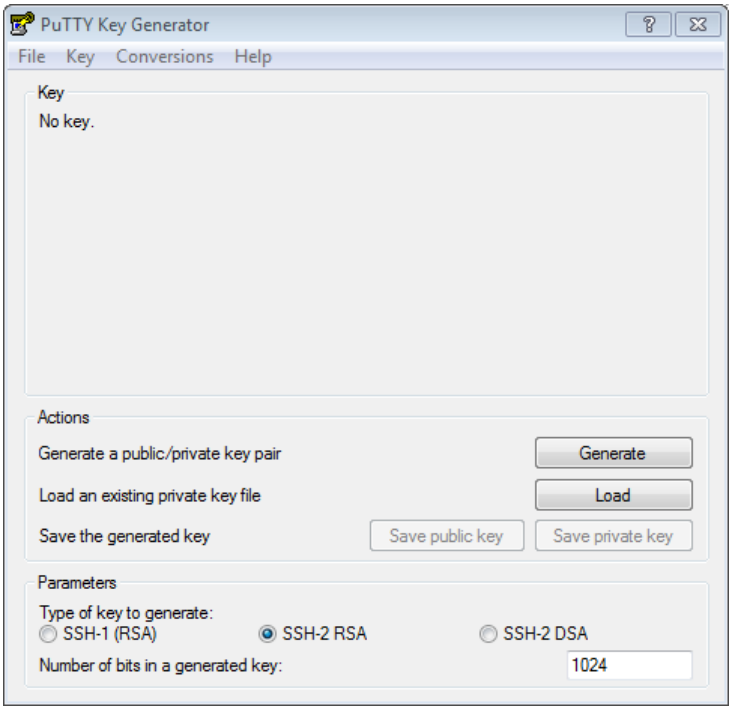
Nous utiliserons le logiciel PuTTy sur Windows. Ce logiciel est gratuit et très répandu. Lorsqu'on le télécharge, on récupère un ensemble d'outils associés qui seront nécessaires pour utiliser le « FILENAME SFTP ».

Voici une copie d'écran des outils disponibles lorsque l'on télécharge « PuTTy » :

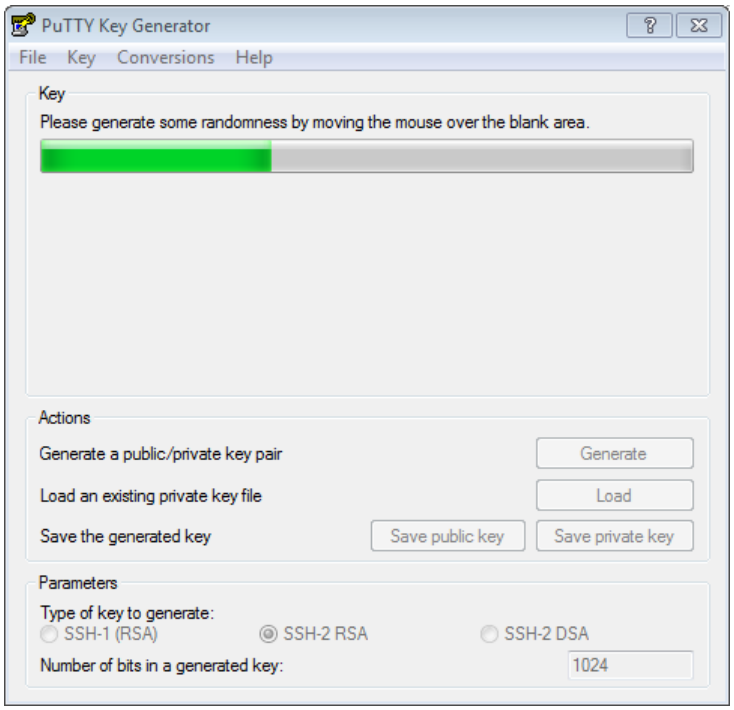


La première action à effectuer est d'exécuter l'outil « PUTTYGEN.EXE » qui permet de générer le couple de clés privée et publique. C'est ce couple de clés qui va nous permettre de nous authentifier auprès du serveur SFTP. **Nous devons toutefois disposer d'un compte utilisateur déclaré sur le serveur SFTP**. C'est avec ce compte que nous nous connecterons lors de l'appel à la méthode « FILENAME SFTP », toutefois nous n'utiliserons pas de mot de passe pour nous authentifier mais le couple clés privée et publique.

Vous avez alors l'écran suivant :

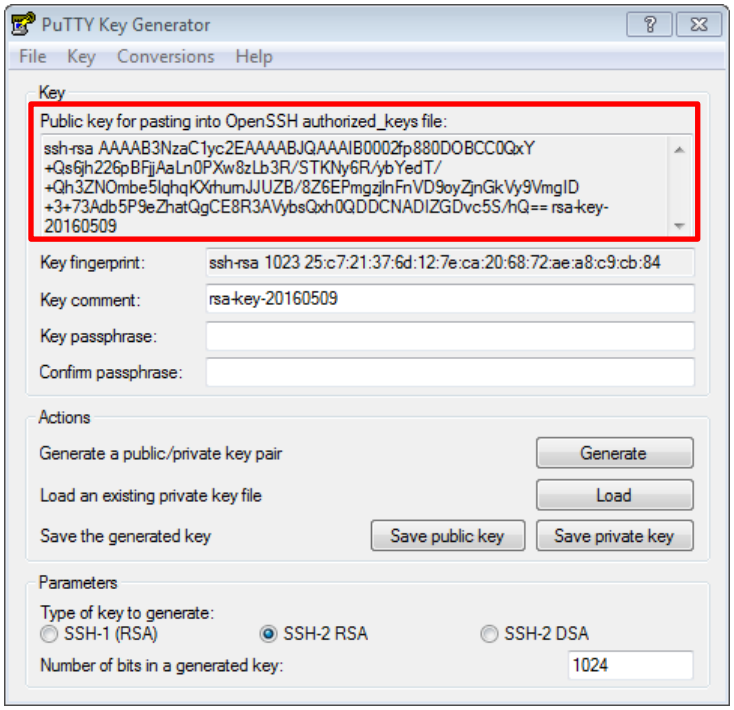


Le choix par défaut proposé est un couple de clés de type RSA sur 1024 bits, ce qui est correct. Nous cliquons ensuite sur le bouton « Generate », ce qui affiche la fenêtre suivante :



Il nous est demandé de « générer de l'aléatoire » en bougeant le pointeur de souris pour construire des clés suffisamment « robustes ». Cette manipulation peut prendre un peu de temps.

Une fois cela effectué, nous obtenons l'écran suivant :



Il est alors **fortement recommandé** de protéger la clé privée par une phrase secrète dite « passphrase » afin d'éviter que quelqu'un d'autre que nous ne l'utilise. En effet la clé privée est le moyen qui va nous permettre de nous authentifier auprès du serveur SFTP.
Il faut alors sauvegarder la clé privée et la clé publique dans des fichiers. Le fichier de la clé privée est à protéger (stocké dans un répertoire du poste que seul notre compte peut accéder).

La clé privée restera sur notre poste et sera utilisée par l'outil « PuTTy agent » tandis que la clé publique sera placée sur le serveur SFTP.

Le détail de cette clé publique est encadré en rouge dans la copie d'écran précédente. Cette zone est du simple texte et peut être sélectionnée afin d'être copiée dans le presse papiers. Cela permet de coller ensuite très facilement ces informations dans un fichier appelé « authorized_keys » se trouvant sur le serveur SFTP dans le répertoire « .ssh » du compte utilisateur déclaré sur le serveur. Dans notre exemple voici la partie exacte à copier :

```
ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAAIB0002fp880DOBCC0QxY+Qs6jh226pBFjjAaLn0PXw8zLb3R/STKNy6R/ybYe
dT/+Qh3ZNOmbe5IqhqKXrhurnJJUZB/8Z6EPmgzjlnFnVD9oyZjnGkVy9VmgID+3+73Adb5P9eZhatQgCE8R3A
VybsQxh0QDDCNADIZGDvc5S/hQ==
```

Il ne faut pas prendre la fin de chaîne située après les 2 signes « = » car c'est un commentaire descriptif de la clé, toujours dans notre cas c'est la chaîne :

```
rsa-key-20160509
```

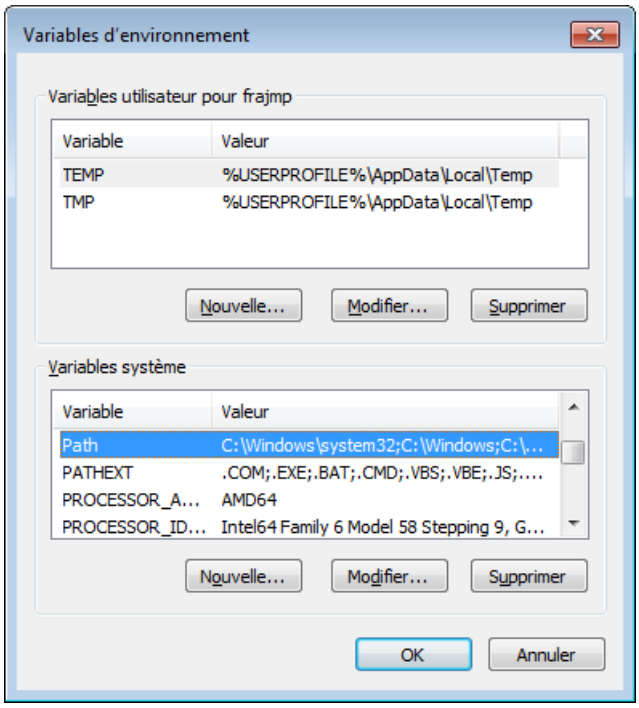
Sur le serveur SFTP, connecté avec le compte utilisateur déclaré, nous créons le répertoire caché « .ssh » dans le répertoire « home » du compte, s'il n'existe pas déjà :

```
cd ~
mkdir .ssh
```

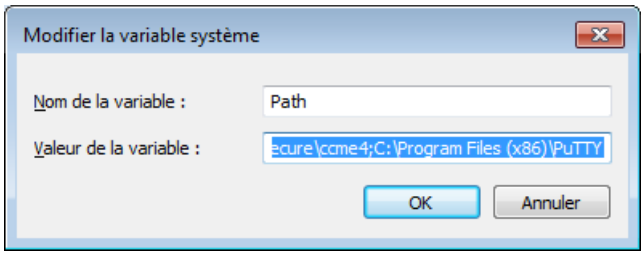
Puis dans ce répertoire « .ssh », s'il n'existe pas, nous créons le fichier « authorized_keys » en collant le texte copié et en sauvegardant ensuite ce fichier :

```
cd .ssh
vi authorized_keys
```

Dernier point important, la méthode « FILENAME SFTP » va utiliser un client SFTP et nécessiter que l'outil « psftp.exe » récupéré avec PuTTy soit accessible. Pour cela le répertoire contenant ce programme doit être ajouté dans la variable système « PATH ». Concrètement il faut aller dans « Panneau de configuration » puis « Système et sécurité » et enfin « Système », ensuite cliquer sur le lien « Paramètres système avancés ». Dans la fenêtre on clique alors sur le bouton « Variables d'environnement… ». La variable « PATH » est une variable système :



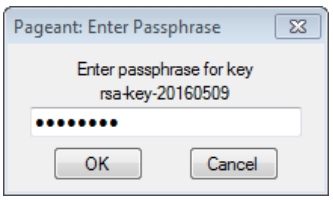
Après sélection de cette variable on clique sur « Modifier… » et on ajoute le chemin à la fin (dans notre cas « C:\Program Files (x86)\PuTTY », en le séparant du précédent chemin avec un « ; ») :



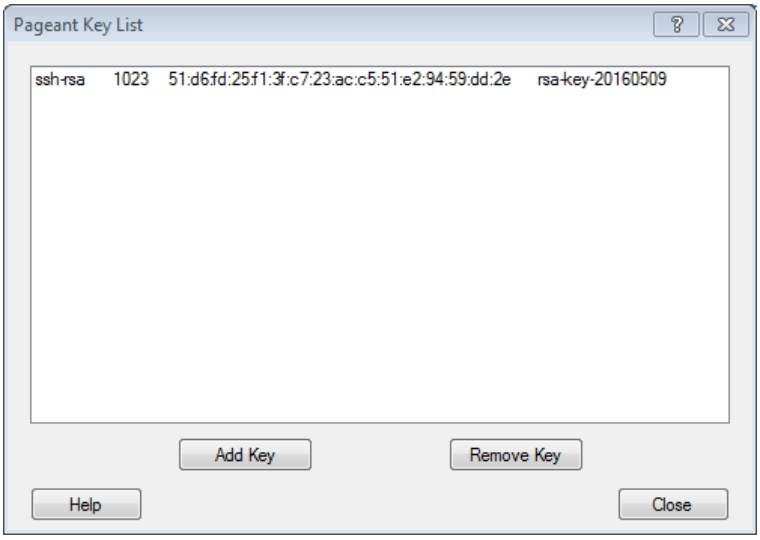
Les étapes préparatoires sont alors terminées.

2.2.2. Utilisation

Sur notre poste nous lançons l'outil « PuTTy agent ». Par défaut celui-ci se place en « minimisé » près de l'heure système. Il faut, via un clic droit sur son icône, faire « Add key ». Il demande alors d'indiquer l'emplacement du fichier de notre clé privée. Si nous avons protégé celui-ci par une phrase secrète, il nous demande de l'indiquer comme suit :



La clé est maintenant chargée, ce qui va permettre à l'agent de nous authentifier sur le serveur SFTP via notre compte utilisateur :



Dans notre session SAS, nous pouvons maintenant accéder au serveur SFTP. Le code suivant récupère le fichier texte « setinit.sas » et le place dans la table TEST de la bibliothèque WORK. Comme nous lançons la session depuis Windows, il est nécessaire de mentionner le compte utilisateur via le paramètre USER (c'est avec ce compte que nous avons associé les clés privée et publique). Enfin nous avons mis l'option DEBUG qui permet d'avoir plus de trace dans le journal d'exécution :

```
filename inf sftp '/tmp/setinit.sas' host="aix2" user="frajmp" debug;
data test;
  infile inf truncover;
  input ligne $255.;
run;

proc print data=test;
run;
```

Concrètement la méthode va appeler le client « psftp.exe » en lui passant le compte utilisateur mentionné et le serveur SFTP (paramètre HOST). L'outil « psftp.exe » va établir la connexion en utilisant la clé privée chargée dans le « PuTTy agent ». SAS va ensuite pouvoir accéder aux données via cette connexion.

3. LIENS UTILES

La documentation pour la méthode « FILENAME SFTP » avec de nombreux exemples illustrant ses possibilités :
FILENAME SFTP

La note technique pour implémenter la méthode SFTP :
http://support.sas.com/techsup/technote/ts800.pdf
Cette note technique décrit notamment comment procéder depuis un client Unix/Linux.

4. CONCLUSION

Les deux méthodes d'accès détaillées dans cet article permettent d'accéder et d'exploiter des données distantes hétérogènes et sous divers formats dans SAS, ce qui étend ses capacités.

Jean-Marie POILANE
Consultant Support Clients SAS France