Documentation EMu

La fonction Après l'exportation

Document Version 1.1

EMu Version 4.0

Contents

SECTION 1

Vue d’ensemble

La fonction Exports Planifiés introduite avec EMu 4.0.02 fournit un mécanisme pour l'exportation des données hors d’EMu de façon régulière ou ad hoc. Les Exports Planifiés sont exécutés par le serveur sans aucune intervention de l'utilisateur. Lorsqu'une exportation est terminée, un enregistrement est ajouté au module Exports avec :

- les résultats de l’exportation
- une liste des fichiers générés
- toute erreur associée

L'utilisateur doit alors récupérer l'enregistrement à partir du module Exports pour accéder aux résultats et aux données exportées.

Introduit avec EMu 4.0.04, la fonction Après l’exportation permet à une commande d’être exécutée une fois l'exportation terminée. Entre autres choses, la commande peut :

- Envoyer par email les fichiers d'exportation à une liste d'utilisateurs.
- Envoyer par email les résultats de l'exportation à une liste d'utilisateurs.
- Copier les fichiers d'exportation vers une autre machine derrière un firewall sécurisé.
- Copier les fichiers d'exportation sur l'internet via un mécanisme de transfert sécurisé.
- Envoyer un texto à une liste de numéros de téléphone.

En fait, une commande Après l’exportation peut effectuer n’importe quelle tâche comme elle a un accès complet à l’enregistrement Exports généré. La commande est exécutée sur le serveur EMu permettant l'accès à toutes les fonctions offertes par le serveur. La fonction Après l’exportation est conçue pour permettre d’ajouter de nouvelles commandes dans le framework existant. Afin de simplifier la création de nouvelles commandes, le module perl `KE::Export` est fourni, il intègre la plupart des fonctionnalités requises par une commande Après l’exportation.

SECTION 2

Définir une commande Après l'exportation

Un export planifié est configuré en utilisant la boîte de dialogue Propriétés de l'exportation dans un module (en sélectionnant **Outils>Export** dans la barre de menu du module). Une commande Après l'exportation est ajoutée à un export planifié en utilisant l'onglet Après l'exportation de la boîte de dialogue Propriétés de l'exportation :

- si une commande Après l'exportation a été définie, elle peut être sélectionnée dans la liste déroulante *Commande* sur l'onglet Après l'exportation.
- une commande peut nécessiter d'avoir des valeurs fournies par un utilisateur afin d'être exécutée, dans ce cas les champs de saisie pour les valeurs requises s'affichent sur l'onglet Après l'exportation lorsque la commande est sélectionnée dans la liste déroulante *Commande*.

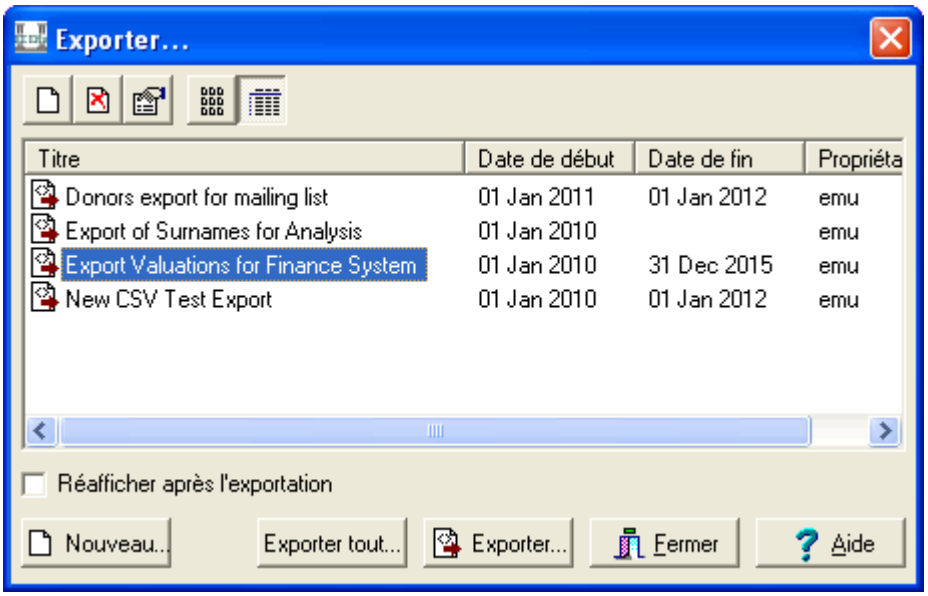
Dans EMu :

1. Ouvrir un module.
2. Rechercher ou lister d'une autre façon un groupe d'enregistrements.
3. Sélectionner **Outils>Export** dans la barre de menu

   -OU-

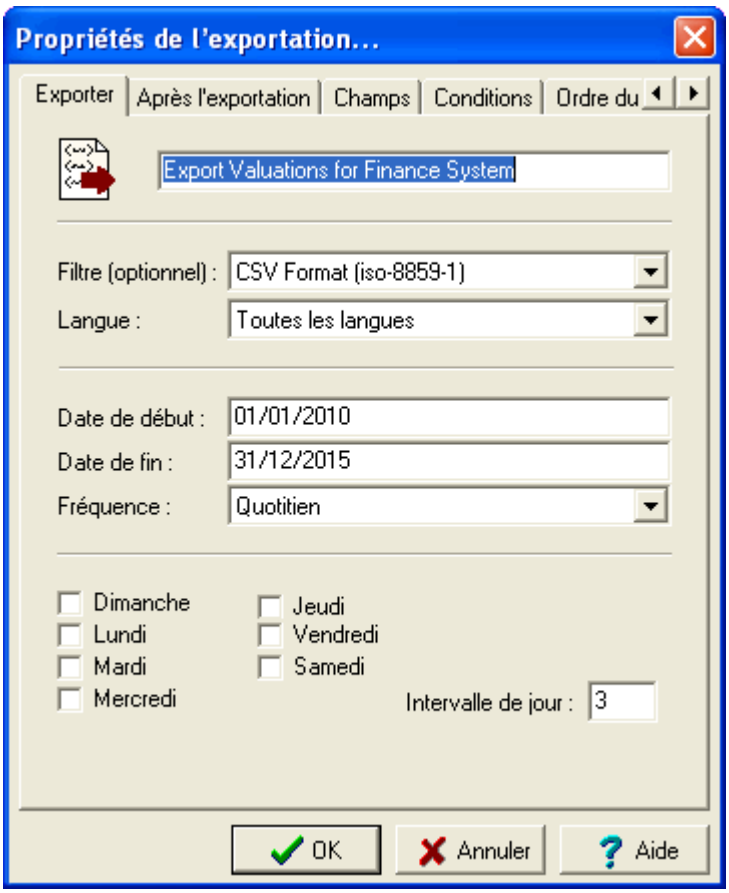
   Utiliser le raccourci clavier, ALT+T+X.

   La boîte de dialogue Exports s'affiche avec une liste d'exports planifiés pour le module en cours :

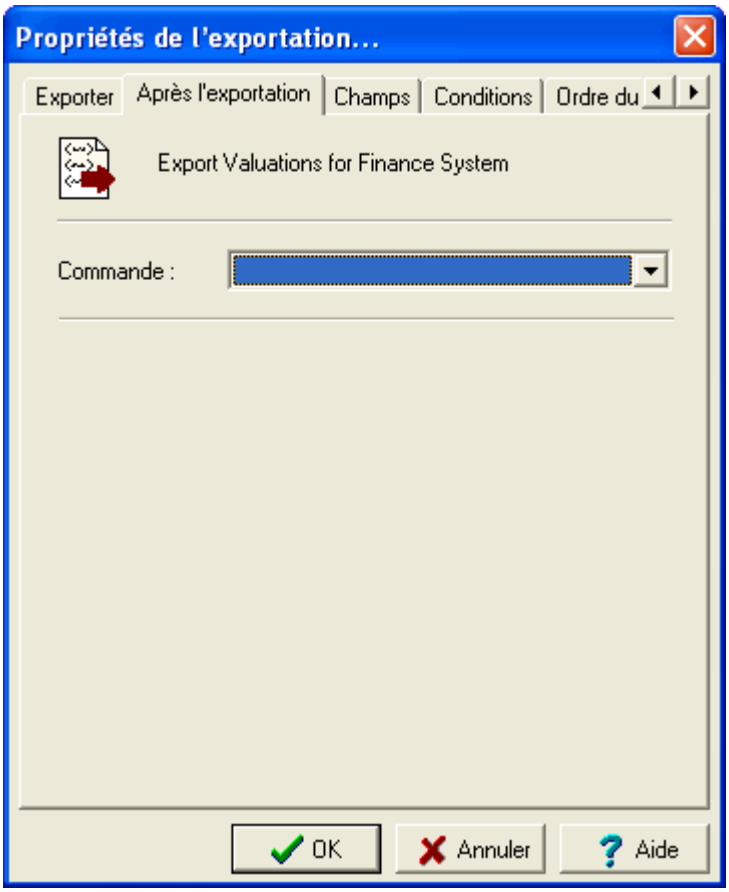


4. Sélectionner un export planifié et cliquer  .

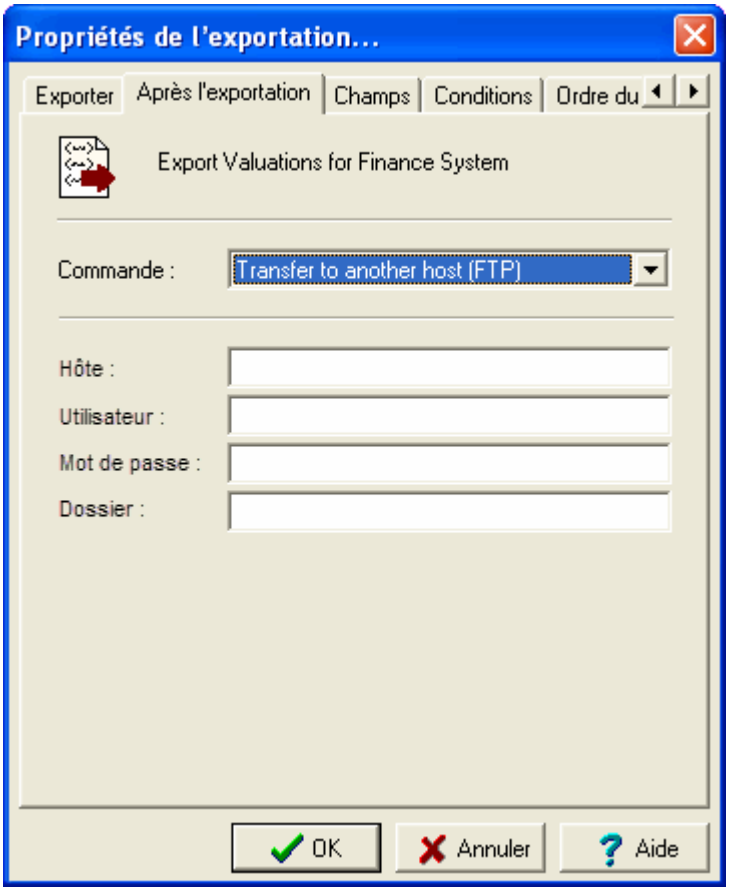
La boîte de dialogue Propriétés de l'exportation s'affiche :



5. Sélectionner l'onglet **Après l'exportation** :

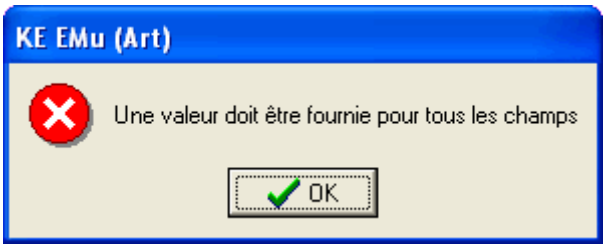


6. Dans la liste déroulante *Commande*, sélectionner la commande à exécuter.
   Si la commande a besoin de valeurs avant d'être lancée, des champs de saisie pour les paramètres s'afficheront dans l'onglet Après l'exportation :



7. Saisir une valeur pour chacun des champs et cliquer 

.
Si un paramètre n'a pas été fourni, un message d'erreur s'affiche :



Si tous les paramètres ont une valeur, la commande Après l'exportation est enregistrée et sera exécutée lorsque le prochain export planifié est lancé.

SECTION 3

Développer un script Après l'exportation

Le script Après l'exportation

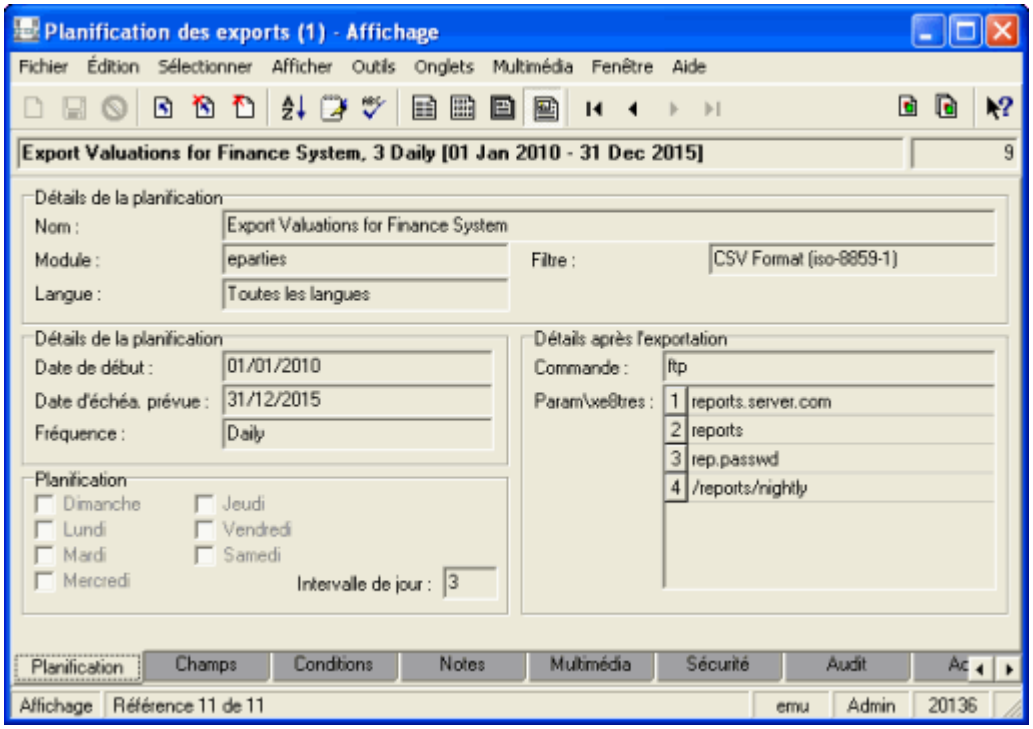
Le module Planification des exports contient un enregistrement pour chaque export planifié défini dans chaque module.

Quand une commande Après l'exportation est ajoutée à un export planifié, un script situé sur le serveur EMu est exécuté après l'export planifié. Dans l'enregistrement Planification des exports pour l'export planifié :

- *Commande : (Détails Après l'exportation)* contient le nom du script à exécuter.
- *Paramètres : (Détails Après l'exportation)* liste les paramètres requis par le script.

Dans l'enregistrement Planification des exports ci-dessous, le script exécuté quand l'export planifié est complété est appelé `ftp`. Il a quatre paramètres :

- `reports.server.com` (nom d'hôte)
- `reports` (nom d'utilisateur)
- `rep.passwd` (mot de passe)
- `/reports/nightly` (localisation)



Le fichier de script est situé dans un des répertoires suivants sur le serveur EMu :

- `local/etc/exports/after/`*table*
- `local/etc/exports/after`
- `etc/exports/after/`*table*
- `etc/exports/after`

où *table* est le nom de la table (module) sur lequel l'exportation est effectuée. Le nom de la table est indiqué dans le champ *Module : (Détails de la planification)* (comme indiqué ci-dessus). Pour localiser le script, le système recherche un fichier appelé `ftp` (dans ce cas) dans chacun des répertoires listés, du premier au dernier. Lorsque le fichier est trouvé, le script stocké dedans est exécuté.

La hiérarchie des répertoires listés ci-dessus permet d'ajouter des versions personnalisées de scripts existants en plaçant simplement un fichier avec le même nom que le script existant dans l'un des répertoires `local`.

Un script Après l'exportation est invoqué par l'une de deux méthodes :

1. Dans la première méthode, un script est invoqué quand un utilisateur sélectionne une commande dans la liste déroulante *Commande* de l'onglet Après l'exportation de la boîte de dialogue Propriétés de l'exportation.

   Dans ce cas, le client EMu invoque le script pour obtenir le titre et la liste des paramètres requis par la commande : le titre s'affiche dans la liste déroulante *Commande* et les paramètres sont affichés en dessous. Le client EMu invoque le script avec un argument, une option indiquant la langue à utiliser pour afficher le titre du script et les paramètres. L'utilisation est :

   ```
   script -l num
   ```

   où *num* est un nombre correspondant à la langue à utiliser :

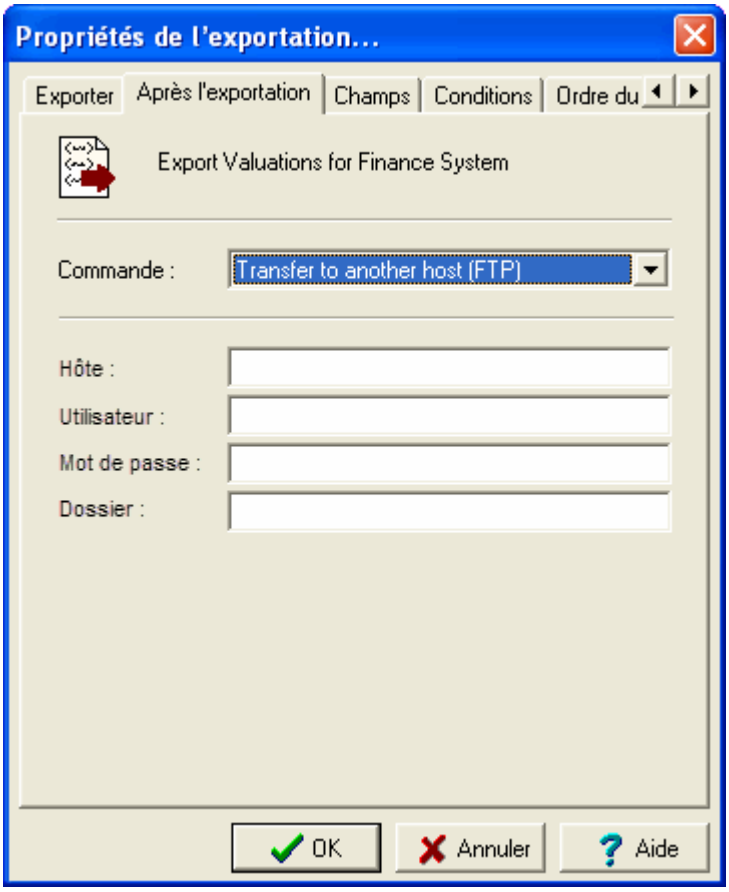
Nombre	Langue
0	Anglais
1	Français
2	Anglais (Américain)
3	Espagnol
4	Allemand
5	Italien
6	Hollandais
7	Danois
8	Polonais
9	Norvégien
10	Suédois
11	Grec
12	Arabe
13	Hébreu
14	Français (Canadien)
15	Finlandais

 Le `l` dans le `script -l`*num* est un L minuscule.

Par exemple, `ftp - l0` produit :

```
Transfer to another host (FTP)
Host:
User:
Password:
Folder:
```

La première ligne est affichée dans la liste déroulante *Commande* et les lignes restantes sont affichées comme étiquettes pour les champs de saisie :



Si la commande n'est pas supportée par la machine locale, alors aucun output ne devrait être généré (la commande doit être cachée dans la liste déroulante *Commande*) et un statut de sortie non nul retourné.

2. Dans la seconde méthode, la commande backend `emuexport` invoque le script Après l'exportation après qu'un export planifié ait eu lieu et qu'un enregistrement des résultats de l'exportation ait été créé dans le module Exports. Dans ce cas, le script est invoqué avec le NEI (Numéro d'Enregistrement Interne), c'est à dire l'irn (Internal Record Number), de l'enregistrement créé dans le module Exports (eexports). L'utilisation est :

   `script` *exportirn*

   Le script doit utiliser *exportirn* pour accéder à l'enregistrement Exports (eexports) et effectuer les activités pour lesquelles le script a été conçu (par exemple, envoyer un mail, copier des fichiers, etc.). Si une erreur survient pendant le traitement de l'exportation, un message d'erreur devrait être imprimé et un statut de sortie non nul retourné. Si le script se termine avec succès, un statut de sortie zéro est retourné.

Ces deux méthodes sont expliquées en détail dans *Créer un script Après l'exportation* (page 13).

Le code perl ci-dessous est un échantillon du script `ftp` :

```
#!/usr/bin/perl

#
#  Copyright (c) 1998-2011 KE Software Pty Ltd
#

use strict;
use KE::Export;

#
#  Parameters for ftp.
#
my $prompts =
{
        0       => [
                        "Transfer to another host (FTP)",
                        "Host:",
                        "User:",
                        "Password:",
                        "Folder:"
                   ]
};

#
#  Check whether ftp is supported on this machine.
#
my $ftp = KE::Export::Ftp->new();
if (! $ftp->IsSupported())
{
        exit 1;
}

#
#  Parse the arguments.
#
my $export = KE::Export->new();
if (! $export->ParseArgs(\@ARGV))
{
        exit 1;
}

#
#  List parameters if required.
#
if ($export->ListParameters($prompts))
{
        exit 0;
}

#
#  Do the transfer with the required arguments.
#
my $status = $ftp->Execute
(
        host            =>      $export->GetData('Parameters_tab')->[0],
        user            =>      $export->GetData('Parameters_tab')->[1],
        password        =>      $export->GetData('Parameters_tab')->[2],
        destination     =>      $export->GetData('Parameters_tab')->[3],
        filelist        =>      $export->GetData('FileName_tab')
```

```
);

#
#  Send back the error status.
#
exit $status;
```

Ce script utilise le module perl `KE::Export` qui fournit la plupart des fonctionnalités requises.

En gros le script :

1. vérifie si le serveur fournit un support FTP. Sinon, il retourne un statut de sortie non nul (c'est à dire `1`).
2. analyse les arguments pour déterminer quelle version du script a été invoquée.

   Si les arguments ne sont pas valides, un statut de sortie non nul est retourné.

   -OU-

   Si un argument valide `-l`*num* a été donné et le script a été invoqué en utilisant la première méthode décrite ci-dessus, le script affiche le titre et les paramètres pour la langue *num*. Puis se termine avec un statut de sortie zéro (indiquant la réussite).

   -OU-

   Si le script a été invoqué en utilisant la deuxième méthode décrite ci-dessus, le transfert des fichiers est effectué en utilisant ftp. Un objet ftp, passé les paramètres requis, transfert les fichiers. Le statut de l'objet ftp est utilisé pour le statut de sortie, où une valeur non nulle indique que le transfert a échoué, et une valeur de zéro indique le succès du transfert.

Pour une description complète des fonctionnalités fournies par le module `KE::Export`, voir *Utilisation de KE::Export*.

Créer un script Après l'exportation

Le script pour une commande Après l'exportation doit gérer les deux cas d'utilisation où le script est invoqué avec :

- `-l`*num* pour fournir le titre de la commande et la liste des paramètres

  -OU-
- avec le NEI d'un enregistrement eexports pour effectuer ce que le script est censé de faire

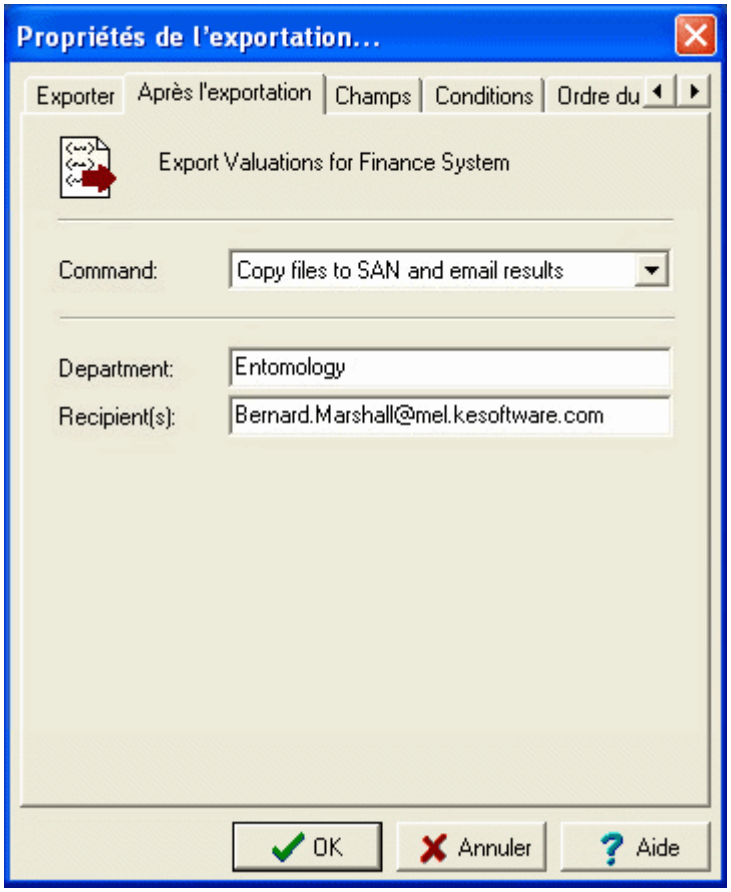
Afin de rendre l'écriture du script plus facile, un module perl `KE::Export` est fourni qui englobe la plupart des fonctionnalités requises par un script Après l'exportation. Il est recommandé que vous utilisiez le module pour réduire le temps de développement. Pour une description complète des fonctionnalités fournies par le module `KE::Export`, voir *Utilisation de KE::Export*.

Pour démontrer comment une nouvelle commande peut être créée, nous allons écrire un script qui copie les fichiers de sortie pour une exportation vers un emplacement sur un disque SAN (`/exports`) basé sur la date de l'exportation et un département spécifié par l'utilisateur, et ensuite envoyer un email à une adresse fournie par l'utilisateur.

La première partie du script implique la mise en place du titre et des paramètres. Deux paramètres sont requis, le premier est le département sous lequel déposer les fichiers d'exportation et le second est les adresses email à notifier une fois que les fichiers ont été copiés :

```
#
#  Parameters for copy and email notification
#
my $prompts =
{
        0       => [
                        "Copy files to SAN and email results",
                        "Department:",
                        "Recipient(s):"
                    ]
};
```

La première chaîne de caractères est le titre à afficher dans la liste déroulante *Commande*, et les deux paramètres suivants permettent de saisir le département et les adresses email. L'onglet Après l'exportation en résultant est :



Nous vérifions ensuite que le serveur supporte la copie des fichiers et l'envoi de messages par emails. Si le support pour l'un deux n'est pas fourni, la commande doit être cachée dans la liste déroulante *Commande*. Un statut de sortie de `1` indique que la commande n'est pas supportée :

```
#
#  Check whether email and copy are supported on this machine.
#
my $copy = KE::Export::Copy->new();
my $email = KE::Export::Email->new();
if (! $email->IsSupported() || ! $copy->IsSupported())
{
        exit 1;
}
```

Lorsque nous avons confirmé que la fonctionnalité requise est supportée, nous vérifions que les arguments fournis sont valides. S'ils ne le sont pas, la sortie a un statut d'erreur `1` :

```
#
#  Parse the arguments.
#
my $export = KE::Export->new();
if (! $export->ParseArgs(\@ARGV))
{
        exit 1;
}
```

Puisque les arguments fournis sont acceptables, nous vérifions `-l`*num*. Si les arguments correspondent, les étiquettes définies ci-dessous sont affichées et la sortie a un statut de réussite de `0` :

```
#
#  List parameters if required.
#
if ($export->ListParameters($prompts))
{
        exit 0;
}
```

Si nous en arrivons là, nous devons traiter l'enregistrement `eexports` dont le NEI a été fourni comme argument. D'abord nous construisons le chemin complet vers le dossier dans lequel nous voulons stocker les fichiers d'exportation. Nous utilisons la colonne *FileRunDate* de l'enregistrement eexports pour obtenir la date à laquelle l'exportation a été exécutée, et la première entrée dans la liste des colonnes *Parameters_tab* pour obtenir le département saisi par l'utilisateur. Une fois que nous avons le chemin de destination, nous vérifions que le dossier existe (en utilisant `mkdir`). Si le dossier ne peut pas être créé, la sortie a un statut d'erreur de `1` :

```
#
#  Build up the destination directory and make sure it exists.
#
my $date = $export->GetData('FileRunDate');
my $department = $export->GetData('Parameters_tab')->[0];
my $destination = "/tmp/$department/$date";
if (system("mkdir -p '$destination'") != 0)
{
        exit 1;
}
```

Il est temps maintenant de copier les fichiers d'exportation vers le dossier de destination. Nous utilisons un objet `KE::Export::Copy` pour effectuer le transfert. La colonne *FileName_tab* contient une liste de tous les fichiers créés par l'exportation :

```
#
#  Do the copy with the required arguments.
#
my $status = $copy->Execute
(
        destination     =>      $destination,
        filelist        =>      $export->GetData('FileName_tab')
);
```

Maintenant nous construisons le corps du message email à envoyer aux destinataires. Nous incluons le nom de l'exportation, la date à laquelle elle a été effectuée, le dossier dans lequel les fichiers d'exportation ont été copiés et indiquons si le transfert a réussi :

```
#
#  Build up the email message to send
#
my $body = "The files from export \"" .
           $export->GetData('ScheduleRef:eschedule:Name') .
           "\", run on $date have been copied to $destination.\n" .
           "The copy " . $status ? "failed" : "succeeded.\n";
```

Enfin, nous envoyons le message email à tous les destinataires. Nous utilisons le second paramètre pour extraire les adresses email des destinataires. Le sujet de l'email est le nom de l'export planifié :

```
#
#  Do the email with the required arguments.
#
my $status = $email->Execute
(
    recipients   =>   $export->GetData('Parameters_tab')->[1],
    subject      =>   $export->GetData('ScheduleRef:eschedule:Name'),
    body         =>   $body
);
```

Nous retournons le statut de l'objet email, indiquant si les emails ont été envoyés correctement ou non :

```
#
#  Send back the error status.
#
exit $status;
```

Le script complet est :

```
#!/usr/bin/perl

#
#  Copyright (c) 1998-2011 KE Software Pty Ltd
#

use strict;
use KE::Export;

#
#  Parameters for copy and email notification
#
my $prompts =
{
        0       => [
                        "Copy files to SAN and email results",
                        "Department:",
                        "Recipient(s):"
                   ]
};

#
#  Check whether email and copy are supported on this machine.
#
my $copy = KE::Export::Copy->new();
my $email = KE::Export::Email->new();
if (! $email->IsSupported() || ! $copy->IsSupported())
{
        exit 1;
}

#
#  Parse the arguments.
#
my $export = KE::Export->new();
if (! $export->ParseArgs(\@ARGV))
{
        exit 1;
}
```

```
#
#  List parameters if required.
#
if ($export->ListParameters($prompts))
{
        exit 0;
}

#
#  Build up the destination directory and make sure it exists.
#
my $date = $export->GetData('FileRunDate');
my $department = $export->GetData('Parameters_tab')->[0];
my $destination = "/tmp/$department/$date";
if (system("mkdir -p '$destination'") != 0)
{
        exit 1;
}

#
#  Do the copy with the required arguments.
#
my $status = $copy->Execute
(
    destination   =>    $destination,
    filelist      =>    $export->GetData('FileName_tab')
);

#
#  Build up the email message to send.
#
my $body = "The files from export \"" .
        $export->GetData('ScheduleRef:eschedule:Name') .
        "\", run on $date have been copied to $destination.\n" .
        "The copy " . ($status ? "failed" : "succeeded") . ".\n";

#
#  Do the email with the required arguments.
#
my $status = $email->Execute
(
     recipients    =>    $export->GetData('Parameters_tab')->[1],
     subject       =>    $export->GetData('ScheduleRef:eschedule:Name'),
     body          =>    $body
);

#
#  Send back the error status.
#
exit $status;
```

Le script doit être placé dans le répertoire `local/etc/exports/after` puisque c'est un script personnalisé. Le nom de fichier du script n'est pas important, mais quelque chose comme `copyemail` serait approprié. N'oubliez pas de changer les permissions du fichier de sorte qu'il puisse être exécuté (c'est à dire `chmod 755 copyemail`). Votre script est maintenant prêt à l'emploi.

Utilisation de KE::Export

Le paquet `KE::Export` fournit un certain nombre d'objets très utiles qui simplifient le processus de création d'un script Après l'exportation. Le fichier du paquet est dans `utils/KE/Export.pm` et est complètement documenté. Pour consulter la documentation, utiliser `pod2text Export.pm` (en supposant que vous êtes dans le répertoire `utils/KE`). Pour votre commodité la documentation est reproduite ici :

```
NOM
       KE::Export - Un ensemble d'objets utilisables par les scripts Après
       l'exportation
SYNOPSIS
    use KE::Export;
    my $prompts =
      {
         0    => [
                      'Copy to another folder (CP)',
                      'Folder:'
                 ]
      };

      my $copy = KE::Export::Copy->new();
      if (! $copy->IsSupported())
      {
            exit 1;
      }

      my $export = KE::Export->new();
      if (! $export->ParseArgs(\@ARGV))
      {
            exit 1;
      }
      if ($export->ListParameters($prompts))
      {
            exit 0;
      }

      my $status = $copy->Execute
      (
         destination  =>   $export->GetData('Parameters_tab')->[0],
         filelist     =>   $export->GetData('FileName_tab')
      );

      exit $status;
DESCRIPTION
       Le module "KE::Export" fournit un ensemble d'objets pour rendre la
       mise en œuvre des commandes Après l'exportation plus facile. Une
       commande Après l'exportation peut être enregistrée avec un export
       planifié via l'onglet "Après l'exportation" dans la boîte de
       dialogue Propriétés de l'exportation. Si une commande Après
       l'exportation est enregistrée avec un export planifié, la commande
```

est exécutée une fois la phase d'exportation terminée.

La commande Après l'exportation fournit un mécanisme pour traiter les données exportées après qu'elles aient été générées. En particulier la commande peut :

* Envoyer par email les fichiers d'exportation à une liste d'utilisateurs.

* Envoyer par email les résultats de l'exportation à une liste d'utilisateurs.

* Copier les fichiers d'exportation vers une autre machine.

* Envoyer un texto à une liste de numéros de téléphone.

Une commande Après l'exportation correspond à un script situé dans un des répertoires suivants sur la machine serveur :

local/etc/exports/I<table>/after
local/etc/exports/after
etc/exports/I<table>/after
etc/exports/after

Les répertoires sont examinés dans l'ordre spécifié ci-dessus pour localiser le script requis. En utilisant ce mécanisme, il est possible de remplacer un script fourni avec le système avec un script construit sur mesure. Pour ce faire il suffit d'ajouter votre script personnalisé, avec le même nom que le script que vous remplacez, dans un répertoire figurant au-dessus du répertoire dans lequel le script système est situé. Par exemple, si vous voulez remplacer le script système "ftp" situé dans etc/export/after/ftp, vous devez placer votre script dans le fichier local/etc/export/after/ftp.

Un script Après l'exportation est invoqué par l'une de deux méthodes :

"script -l*num*"

où *num* est le numéro de la langue à utiliser pour l'output des paramètres. Cette forme du script imprime le *titre* du script à utiliser dans la liste déroulante de l'onglet "Après l'exportation" du client et une liste des étiquettes des paramètres requis, un par ligne. Par exemple, le script "ftp" invoqué par "ftp -l0" (pour imprimer le titre et les paramètres en anglais) produit :

Transfer to another host (FTP)

Host:

User:

Password:

où la première ligne est affichée dans la liste déroulante "Commande :" de l'onglet "Après l'exportation" de Propriétés de l'exportation et les lignes restantes sont affichées comme champs de saisie sous la liste déroulante "Commande :" avec le texte utilisé comme étiquette. L'utilisateur doit spécifier chacun des paramètres avant qu'une commande Après l'exportation valide puisse être sauvegardée.

Le numéro de la langue fourni via l'option "-l" détermine la langue dans laquelle le output doit apparaître. Les numéros de

langues sont :

0 - Anglais

1 - Français

2 - Anglais (Américain)

3 - Espagnol

4 - Allemand

5 - Italien

6 - Hollandais

7 - Danois

8 - Polonais

9 - Norvégien

10 - Suédois

11 - Grec

12 - Arabe

13 - Hébreu

14 - Français (Canadien)

15 - Finlandais

"script *exportirn*"

La deuxième méthode pour invoquer un script est de fournir le NEI (Numéro d'Enregistrement Interne), c'est à dire l'irn (Internal Record Number) de l'enregistrement dans la table "eexports" sur lequel le script doit opérer. Dans ce cas, le script a besoin d'effectuer ce qui est considéré comme son devoir. Par exemple, dans le cas du script "ftp", les fichiers d'exportation produits seront envoyés par FTP à un autre hôte. Le nom d'utilisateur, le mot de passe et la destination sur l'hôte distant sont utilisés pour faire le transfert.

Le cycle de vie normal d'un objet "KE::Export" est :

1 Créer l'objet (via "new()").

2 Analyser les paramètres du script pour déterminer laquelle de deux utilisations du script est appropriée (via "ParseArgs()").

3 Output du titre et des paramètres si le script a été invoqué avec la première utilisation (via "ListParameters()").

4 Extraire les paramètres et exécuter les fonctionnalités requises si le script a été invoqué avec la deuxième utilisation (via "GetData()").

Pour des exemples sur comment utiliser le module "KE::Export", veuillez voir les scripts Après l'exportation installés sur la machine serveur avec l'installation par défaut.

KE::Export

Un objet "KE::Export" fournit un wrapper autour d'un ensemble de fonctions utilitaires. Ces fonctions sont conçues pour simplifier

le processus d'écriture des scripts Après l'exportation en encapsulant les fonctionnalités standards. En particulier, les fonctions suivantes sont offertes :

* Une manière standard d'analyser les arguments du script afin de déterminer quel type d'invocation a été utilisée. Voir ParseArgs().
* Un mécanisme standard pour l'output des paramètres du script lorsqu'il est invoqué avec l'option "-l". Voir ListParameters().
* Un mécanisme pour déterminer les langues supportées par le serveur. Voir Languages().
* Un moyen d'extraire les données du lot d'enregistrements sous-jacent et de son enregistrement planifié associé. En fait, n'importe quel lien de lot d'enregistrements peut être suivi pour récupérer des données à partir d'autres modules. Voir GetData().

Chaque script Après l'exportation devrait utiliser un objet "KE::Export" pour simplifier l'accès à l'enregistrement export sous-jacent.

Méthodes

new()

```
$export = KE::Batch->new();
```

Crée un objet qui permet d'accéder à l'enregistrement export sous-jacent. L'objet permet également d'accéder aux fonctions d'aide qui simplifient les scripts Après l'exportation. Afin de libérer toutes les ressources associées à un objet "KE::Export" (par exemple, une connexion à un serveur) "undef" devrait être affecté à la variable objet une fois que l'objet n'est plus requis.

GetData($colname)

```
$data = $export->GetData('StartDate');
$host = $export->GetData('Parameters_tab')->[0];
$filelist = $export->GetData('FileList_tab');
$name = $export->GetData('ScheduleRef:eschedule:Name');
```

Récupère les données de la colonne donnée. L'argument $colname peut être le nom de n'importe quelle colonne dans la table "eexports". La valeur retournée est compatible avec le type de données stockées dans la colonne. Une colonne atomique retourne une chaîne de caractères, une table retourne une référence à une liste de chaînes de caractères et une table imbriquée retourne une référence à une liste où chaque élément est lui-même une liste de chaînes de caractères.

Il est également possible d'accéder aux colonnes dans d'autres modules en spécifiant le nom du champ de lien "eexports" suivi du nom de table et de colonne dans la table liée. Chaque composant est séparé par deux points. Il n'y a aucune limite au nombre de composants dans le nom de colonne pour les champs liés. Par exemple, le nom de colonne "ScheduleRef:eschedule:Name" utilise le lien de "eexports" à "eschedule" (via la colonne ScheduleRef) pour accéder au champ Name de la table "eschedule". En d'autres termes, le nom de l'export planifié est récupéré.

La colonne "Parameters_tab" donne accès aux paramètres de commande saisis pour la commande Après l'exportation. La colonne "FileList_tab" fournit une liste de tous les fichiers générés par le processus d'exportation.

Languages()

```
$langlist = $export->Languages();
```

Récupère la liste des langues supportées par le serveur. La liste se compose d'une chaîne de numéros de langue séparés pas des points-virgules. Par exemple, la chaîne "0;1" indique que le serveur supporte l'anglais et le français, où l'anglais est la langue principale. L'appel "Languages()" est utilisé par les commandes Après l'exportation quand du texte est généré par le script. Le texte doit être dans les langues supportées par le serveur.

La fonction "Languages()" retourne la valeur de l'entrée de Registre "System|Setting|Language|Supported".

ListParameters($prompts)

```
my $prompts =
      {
         0    => [
                        "Email export results (SMTP)",
                        "Recipient(s):"
                    ]
      };

      if ($export->ListParameters($prompts))
      {
         exit 0;
      }
```

Imprime le titre et les paramètres pour la commande Après l'exportation. Si l'appel "ParseArgs()"a déterminé la liste des paramètres à imprimer (via l'option "-lnum"), alors le titre et les paramètres pour le numéro de la langue donné sont imprimés et l'appel retourne 1. Si l'option "-lnum" n'a pas été spécifiée, l'appel retourne 0.

L'argument $prompts est une référence à un croisillon, où la clé est le numéro de langue et la valeur est une référence à une liste de chaînes. La première chaîne est le titre et les chaînes suivantes sont les paramètres. La chaîne de titre est affichée dans la liste déroulante Commande, et les chaînes de paramètres sont affichées dessous la liste déroulante Commande avec des champs de saisie de données où les valeurs peuvent être spécifiées.

ParseArgs(\@ARGV)

```
if (! ParseArgs(\@ARGV))
      {
         exit 1;
      }
```

Analyse les arguments pour déterminer si les paramètres doivent être imprimées ("-lnum" a été fourni) ou la commande exécutée (l'irn eexports a été fourni).

Si des options non valides sont trouvées, un message d'utilisation est imprimé et 0 est retourné. Si les arguments

sont corrects, 1 est retourné.

KE::Export::Command

La classe "KE::Export::Command" est la classe de base pour toutes les commandes Après l'exportation. Elle se compose de deux méthodes que chaque commande Après l'exportation doit mettre en œuvre. La première est "IsSupported()" qui retourne 1 si la commande Après l'exportation est supportée par le serveur. Certaines commandes peuvent nécessiter un logiciel spécial ou certains modules perl avant de pouvoir être utilisées. La deuxième méthode est "Execute()",qui exécute la commande elle-même.

Cette classe ne doit pas être appelée directement à partir des scripts Après l'exportation, mais plutôt une sous-classe doit être créée et les méthodes IsSupported() et Execute() réécrites.

Méthodes

new()

```
$export = KE::Batch::Command->new();
$export = KE::Batch::Command->new(debug => 1);
```

Crée un objet utilisé pour exécuter la commande Après l'exportation. La méthode "new()" ne doit pas être appelée directement, utiliser plutôt des versions sous-classes. Le débogage peut être activé en paramétrant l'argument "debug" à une valeur non nulle.

Execute()

```
$status = $command->Execute()
```

Exécute la commande Après l'exportation. La méthode "Execute()" implémente les fonctionnalités requises par la commande Après l'exportation. Par exemple, si la commande doit envoyer par email les fichiers d'exportation à un utilisateur, la méthode doit effectuer l'envoi par email et joindre les fichiers d'exportation.

La méthode retourne 0 si la commande a réussi, sinon 1 est retourné.

Chaque sous-classe doit remplacer cette méthode pour implémenter les fonctionnalités spécifiques à la commande de la sous-classe.

IsSupported()

```
$support = $command->IsSupported();
```

Détermine si la commande Après l'exportation est supportée par le serveur. Une commande Après l'exportation peut dépendre d'un certain nombre de programmes ou de modules perl. La méthode "IsSupported()" vérifie que chaque dépendance est disponible, et si oui retourne 1, sinon 0. Une valeur retournée de 0, supprime la commande Après l'exportation dans la liste déroulante Commande dans le client.

KE::Export::Sftp

La classe "KE::Export::Sftp" permet d'utiliser le transfert de fichiers sécurisé (SFTP) pour copier les fichiers d'exportation vers une autre machine. Les fonctionnalités "sftp" fournies par

l'ensemble de commandes "scp" sont utilisées pour les transferts de fichiers. Le fichier est crypté lors de la copie pour assurer la confidentialité des données.

Méthodes

```
Execute()
      $sftp = KE::Export::Sftp->new();
      $status = $sftp->Execute
      (
         host         =>   'other.machine',
         user         =>   'username',
         password     =>   'passwd',
         destination  =>   '/exports/nightly/',
         filelist     =>   $export->GetData('FileList_tab')
      );
```

Effectue une copie sécurisée des fichiers d'exportation générés vers un autre hôte. Un certain nombre d'arguments sont disponibles, tous sont obligatoires :

host

Le nom d'hôte de la machine sur laquelle les fichiers d'exportation doivent être copiés.

user

Le nom d'utilisateur à utiliser pour vous connecter à la machine distante.

password

Le mot de passe à utiliser pour vous connecter à la machine distante.

destination

Le répertoire dans lequel les fichiers d'exportation doivent être placés. Le répertoire doit exister.

filelist

Une référence à une liste contenant les fichiers à transférer à la machine distante.

"Execute()" retourne 0 si les transferts ont réussi, sinon 1 est retourné. Si une erreur survient, elle est écrite sur stdout.

IsSupported()

```
$sftp = KE::Export::Sftp->new();
$status = $sftp->IsSupported();
```

Indique si le serveur a les dépendances nécessaires installées pour fournir le transfert de fichiers sécurisé. Une valeur retournée de 0 implique que le serveur ne supporte pas le transfert de fichiers sécurisé, tandis qu'une valeur de 1 implique qu'il fournit ce support.

KE::Export::Ftp

La classes "KE::Export::Ftp" permet d'utiliser le transfert de fichiers (FTP) pour copier les fichiers d'exportation vers une autre machine. Les données transférées ne sont pas cryptées pendant le transit. Comme le mot de passe est envoyé au serveur sans

cryptage, c'est à dire en texte clair, "KE::Export::Ftp" doit être utilisé uniquement au sein des réseaux internes, derrière un firewall sécurisé. FTP peut offrir un débit supérieur à SFTP.

Méthodes

```
Execute()
     $ftp = KE::Export::Ftp->new();
     $status = $ftp->Execute
     (
        host         =>   'other.machine',
        user         =>   'username',
        password     =>   'passwd',
        destination  =>   '/exports/nightly/',
        filelist     =>   $export->GetData('FileList_tab')
     );
```

Effectue une copie des fichiers d'exportation générés vers un autre hôte. Un certain nombre d'arguments sont disponibles, tous sont obligatoires :

host

Le nom d'hôte de la machine sur laquelle les fichiers d'exportation doivent être copiés.

user

Le nom d'utilisateur à utiliser pour vous connecter à la machine distante.

password

Le mot de passe à utiliser pour vous connecter à la machine distante.

destination

Le répertoire dans lequel les fichiers d'exportation doivent être placés. Le répertoire doit exister.

filelist

Une référence à une liste contenant les fichiers à transférer à la machine distante.

"Execute()" retourne 0 si les transferts ont réussi, sinon 1 est retourné. Si une erreur survient, elle est écrite sur stdout.

IsSupported()

```
$ftp = KE::Export::Ftp->new();
$status = $ftp->IsSupported();
```

Indique si le serveur a les dépendances nécessaires installées pour fournir le transfert de fichiers. Une valeur retournée de 0 implique que le serveur ne supporte pas le transfert de fichiers, tandis qu'une valeur de 1 implique qu'il fournit ce support.

KE::Export::Scp

La classe "KE::Export::Scp" permet d'utiliser la copie sécurisée (SCP) pour transférer les fichiers d'exportation vers une autre machine. Les données transférées sont cryptées pendant le transit. Une connexion SCP peut être formée avec ou sans un mot de passe. Si un mot de passe n'est pas fourni, les certificats basés sur X509

peuvent être utilisés pour supprimer la nécessité d'un mot de passe. En général, les connexions X509 sont préférées puisqu'un mot de passe n'a pas besoin d'être stocké dans le script de commande.

Méthodes

```
Execute()
      $scp = KE::Export::Scp->new();
      $status = $scp->Execute
      (
         host         =>   'other.machine',
         user         =>   'username',
         password     =>   'passwd',
         destination  =>   '/exports/nightly/',
         filelist     =>   $export->GetData('FileList_tab')
      );
```

Effectue une copie des fichiers d'exportation générés vers un autre hôte. Un certain nombre d'arguments sont disponibles, la plupart sont obligatoires :

host

Le nom d'hôte de la machine sur laquelle les fichiers d'exportation doivent être copiés. Un nom d'hôte doit être fourni.

user

Le nom d'utilisateur à utiliser pour vous connecter à la machine distante. Un nom d'utilisateur doit être fourni.

password

Le mot de passe à utiliser pour vous connecter à la machine distante. Si un mot de passe n'est pas fourni, une connexion certificat X509 est tentée.

destination

Le répertoire dans lequel les fichiers d'exportation doivent être placés. Le répertoire doit exister. Une destination doit être fournie.

filelist

Une référence à une liste contenant les fichiers à transférer à la machine distante. Une liste de fichiers à transférer doit être fournie.

"Execute()" retourne 0 si les transferts ont réussi, sinon 1 est retourné. Si une erreur survient, elle est écrite sur stdout.

IsSupported()

```
$scp = KE::Export::Scp->new();
$status = $scp->IsSupported();
```

Indique si le serveur a les dépendances nécessaires installées pour fournir la copie sécurisée des fichiers. Une valeur retournée de 0 implique que le serveur ne supporte pas la copie des fichiers, tandis qu'une valeur de 1 implique qu'il fournit ce support.

KE::Export::Copy

La classe "KE::Export::Copy" permet de copier les fichiers d'exportation dans un autre endroit sur la même machine. La commande peut également être utilisée pour copier les fichiers d'exportation sur un système de fichiers monté sur la même machine (par exemple un partage SAMBA).

Méthodes

```
Execute()
      $copy = KE::Export::Copy->new();
      $status = $copy->Execute
      (
          destination  =>  '/exports/nightly/',
          filelist     =>  $export->GetData('FileList_tab')
      );
```

Effectue une copie des fichiers d'exportation générés à un autre endroit sur le même hôte. Deux arguments sont disponibles, tous les deux sont obligatoires :

destination

Le répertoire dans lequel les fichiers d'exportation doivent être placés. Le répertoire doit exister.

filelist

Une référence à une liste contenant les fichiers à copier dans un autre endroit.

"Execute()" retourne 0 si la copie a réussi, sinon 1 est retourné. Si une erreur survient, elle est écrite sur stdout.

IsSupported()

```
$copy   = KE::Export::Copy->new();
$status = $copy->IsSupported();
```

Indique si le serveur a les dépendances nécessaires installées pour fournir la copie des fichiers. Une valeur retournée de 0 implique que le serveur ne supporte pas la copie des fichiers, tandis qu'une valeur de 1 implique qu'il fournit ce support.

KE::Export::Email

La classe "KE::Export::Email" permet d'envoyer une notification par email à une liste d'adresses email quand un export planifié est terminé. Si la liste des fichiers d'exportation est fournie, les fichiers sont envoyés en pièce jointe à l'email de notification, sinon seul le résultat de l'exportation est envoyé.

Méthodes

```
Execute()
      $email = KE::Export::Email->new();
      $status = $email->Execute
      (
          recipients   =>   'user1@abc.com, user2@def.com',
          filelist     =>   $export->GetData('FileList_tab')
      );
```

Envoie par email les résultats d'un export planifié à une liste d'utilisateurs. Si les fichiers d'exportation sont fournis, via le paramètre filelist, les fichiers d'exportation sont joints à l'email. Deux arguments sont disponibles, dont l'un est obligatoire :

recipients

Une liste des adresses email séparées par des virgules définissant les utilisateurs devant recevoir les résultats de l'export planifié. Au moins un destinataire doit être fourni.

filelist

Une référence à une liste contenant les fichiers à joindre à l'email. Si une liste de fichiers (filelist) n'est pas fournie, seuls les résultats sont envoyés par email à chaque utilisateur.

"Execute()" retourne 0 si l'email a réussi, sinon 1 est retourné. Si une erreur survient, elle est écrite sur stdout.

IsSupported()

```
$email = KE::Export::Email->new();
$status = $email->IsSupported();
```

Indique si le serveur a les dépendances nécessaires installées pour fournir les services email. Une valeur retournée de 0 implique que le serveur ne supporte pas l'envoi d'email, tandis qu'une valeur de 1 implique qu'il fournit ce support.

Index