GUIDE

STORMSHIELD DATA SECURITY ENTERPRISE

STORMSHIELD DATA CONNECTOR

Chiffrement d'applications métier

Version 10.1

Dernière mise à jour du document : 29 mars 2022

Référence : sds-fr-sd_connector-guide_d_utilisation-v10

Table des matières

Dans la documentation, Stormshield Data Security Enterprise est désigné sous la forme abrégée : SDS.

1. Prise en main

Stormshield Data Connector est un composant fourni avec Stormshield Data Security. Il permet de piloter les autres composants de la Suite.

Stormshield Data Authority Manager ne peut pas être piloté par Stormshield Data Connector.

Stormshield Data Connector comprend :

- Le module PowerShell “Stormshield.DataSecurity.Connector.PowerShell”, qui s'adresse aux administrateurs d'un parc informatique souhaitant faire fonctionner Stormshield Data Security à travers des scripts PowerShell ;
- Les API .NET, qui s'adressent aux développeurs souhaitant intégrer les fonctionnalités de Stormshield Data Security au cœur de leur propre produit.

L'utilisation du composant Stormshield Data Connector nécessite la maîtrise de la programmation PowerShell et/ou d'un langage .NET tel que C#.

Le fonctionnement du module PowerShell ou des API .NET est le même, les commandes utilisées dans les deux cas sont identiques. On appelle une API .NET de la même façon que l'on écrit un appel de cmdlet PowerShell.

Dans ce guide d'utilisation, nous parlons de cmdlet dans le contexte d'un script PowerShell et d'API dans le contexte d'un programme .NET.

2. Installer Stormshield Data Connector

Ce chapitre fournit des informations sur les pré-requis et l'installation de Stormshield Data Connector.

2.1 Configuration requise

Pour connaître la configuration requise sur les systèmes d'exploitation Microsoft, reportez-vous à la section **Compatibilité** de la note de version Stormshield Data Security 10.1.

Pour les systèmes d'exploitation Windows Server, une licence spécifique Serveur est requise pour Stormshield Data Security.

2.2 Installer Stormshield Data Connector

L'installation de Stormshield Data Security est globale. Le produit fourni comprend tous les composants de la suite logicielle. Installez les applications et composants dont vous avez besoin en fonction des droits octroyés par votre clé de licence.

Afin de piloter un composant donné de la Suite avec Stormshield Data Connector, ce composant doit impérativement être installé sur le poste de l'administrateur et de l'utilisateur. Par exemple, si vous souhaitez utiliser les cmdlets ou API du composant Stormshield Data File, installez le composant File sur les postes de travail.

La procédure d'installation est décrite dans le *Guide d'installation et de mise en œuvre* de Stormshield Data Security.

2.3 Résoudre les problèmes

2.3.1 Exception en cas de composant SDS manquant

Si le composant Stormshield Data Security correspondant à une cmdlet/API n'est pas installé sur le poste de travail, une exception est levée. Par exemple, si le composant Stormshield Data File n'est pas installé, la cmdlet `Protect-SDSFile` lève l'exception :

```
Protect-SDSFile : Cette cmdlet dépend du composant Stormshield Data File
mais celui-ci n'est pas installé.
[...]
Protect-SDSFile c:\folder\document.docx
CategoryInfo : NotSpecified: (:) [], ComponentNotInstalledException
FullyQualifiedErrorId :
Stormshield.DataSecurity.Connector.PowerShell.ComponentNotInstalledExcepti
on
```

Dans le cas d'une utilisation de l'API .NET, la même exception est levée.

2.3.2 Chargement du composant Stormshield Data Connector

PowerShell charge automatiquement le composant Stormshield Data Connector à la première utilisation des cmdlets disponibles.

Si ce n'est pas le cas, il faut importer le module avec la commande :

```
Import-Module Stormshield.DataSecurity.Connector.PowerShell
```

3. Utiliser les cmdlets PowerShell

3.1 Utiliser et comprendre les cmdlets

Stormshield Data Connector est fourni avec un module PowerShell prêt à l'emploi, comprenant des cmdlets permettant le pilotage de Stormshield Data Security. Chaque cmdlet permet l'utilisation d'une fonction du produit.

Par exemple, pour récupérer un objet correspondant à l'utilisateur connecté, utilisez la cmdlet suivante dans une console PowerShell :

```
PS C:\> Get-SDSUser

Id                    : asmith
Name                  : Alice Smith
Locked                : False
EmailAddresses        : {asmith@mycompany.com}
EncryptionCertificate : Alice SMITH
SignatureCertificate  : Alice SMITH
```

La liste des cmdlets disponibles est récupérable avec la commande

```
Get-Command -Module Stormshield.DataSecurity.Connector.PowerShell
```

Pour obtenir de l'aide et des exemples sur chaque cmdlet, tapez

```
Get-Help -Full <nom-de-la-cmdlet>
```

Pour plus d'informations, consultez la Liste des cmdlets Stormshield Data Connector.

ASTUCE
Sur un système d'exploitation 64 bits, utilisez une console PowerShell 64 bits.

3.2 Comprendre la politique d'exécution des scripts Windows PowerShell

Windows bloque par défaut l'exécution des scripts PowerShell.

La cmdlet **Get-ExecutionPolicy** permet de savoir quelle est la politique d'exécution en vigueur. Il existe quatre politiques d'exécution Windows PowerShell :

- **Restricted** : Aucun script ne peut être exécuté.
- **AllSigned** : Seuls les scripts signés par un éditeur approuvé peuvent être exécutés.
- **RemoteSigned** : Tous les scripts locaux peuvent être exécutés mais les scripts distants doivent être signés numériquement pour être exécutés.
- **Unrestricted** : Tous les scripts peuvent être exécutés.

La cmdlet **Set-ExecutionPolicy** permet à l'administrateur de modifier l'état d'exécution des scripts PowerShell en cours sur le système. Par exemple, pour exécuter un script PowerShell local, entrez la cmdlet suivante dans une fenêtre PowerShell en tant qu'administrateur :

```
Set-ExecutionPolicy RemoteSigned
```

4. Utiliser les API .NET

Vous pouvez intégrer les API .NET de Stormshield Data Connector à vos applications pour leur permettre d'interagir avec Stormshield Data Security. Dans cette section, les exemples de code sont fournis en utilisant le langage C#.

4.1 Paramétrer un projet Visual Studio

1. Créez un projet .NET C# reposant sur la version 4.5.2 du .NET Framework. Vous devez avoir installé le .NET Framework 4.5.2 Developer Pack.

   Le .NET Framework 4.5.2 Developer Pack n'est nécessaire que pour compiler l'application utilisant Stormshield Data Connector. L'exécution d'une application utilisant Stormshield Data Connector ne le requiert pas.
2. Ajoutez la référence suivante à votre projet : *Stormshield.DataSecurity.Connector*.

Cet assembly se trouve dans le dossier d'installation de Stormshield Data Security : par défaut **C:\Program Files\Arkoon\Security BOX\Connector**.

Vous pourrez alors utiliser l'espace de noms *Stormshield.DataSecurity.Connector* et disposer de l'ensemble des fonctionnalités de l'assembly.

D'autres espaces de noms (présents dans l'assembly *Stormshield.DataSecurity.Connector*) peuvent être également utilisés pour gérer certains objets Stormshield Data Security. Retrouvez ces espaces de noms dans les exemples donnés dans ce document.

Dans certains cas, une DLL d'interopérabilité devra être ajoutée en référence à votre projet. Ces assemblies se trouvent dans le dossier d'installation de Stormshield Data Security.

Sur un système d'exploitation 64 bits, l'exécutable .NET doit être compilé pour la plate-forme 64 bits :

1. Rendez vous dans les propriétés du projet Visual Studio, section "Build".
2. Choisissez "Platform target = x64" ou décochez "Prefer 32-bit" si vous souhaitez rester en "Platform target Any CPU".

4.2 Utiliser les API .NET

Les API .NET Stormshield Data Connector s'utilisent de la même façon que l'ensemble de cmdlets fournies par le module PowerShell. L'espace de noms *Stormshield.DataSecurity.Connector* comprend tous les objets nécessaires pour interagir avec Stormshield Data Security.

Le point d'entrée pour utiliser les API est la classe *Stormshield.DataSecurity.Connector.API*. C'est un objet qui implémente l'interface IDisposable. Une fois l'objet créé, utilisez la méthode *Execute* pour faire appel à une API.

On appelle une API .NET de la même façon qu'on appelle une cmdlet PowerShell. Deux surcharges de la méthode *Execute* sont disponibles.

```
object[] Execute(string)
```

Cette surcharge est utilisée pour appeler une cmdlet PowerShell dont les paramètres ne sont que des chaînes de caractères (objets de type string), et pas des objets .NET. La cmdlet entière (avec ses paramètres) doit être entrée en tant que paramètre string.

```
object[] Execute(string, KeyValuePair<string, object>[])
```

Cette surcharge est utilisée pour appeler une cmdlet PowerShell dont les paramètres peuvent être des objets .NET. Le nom de la cmdlet est entré en tant que premier paramètre. Les paramètres de la cmdlet sont entrés dans le tableau KeyValuePair (reportez-vous au cas d'usage ci-dessous).

Le tableau object[] retourné fonctionne comme suit :

- Il est égal à « null » si l'API ne retourne rien ;
- Il est égal à un tableau pouvant contenir un ou plusieurs objets si l'API retourne un ou plusieurs résultats ;
- Les éléments du tableau ne peuvent jamais être « null ».

4.3 Exemples

4.3.1 Première surcharge de la méthode *Execute*

Cet exemple montre comment savoir quel utilisateur est connecté.

```
using Stormshield.DataSecurity.Connector;
using (API api = new API())
{
  object[] objects = api.Execute("Get-SDSUser");
}
```

Si le tableau retourné n'est pas vide et le premier élément n'est pas « null », l'utilisateur a été retrouvé avec succès. Le premier élément peut alors être converti en classe User depuis l'espace de noms *Stormshield.DataSecurity.Connector.Kernel* .

```
using Stormshield.DataSecurity.Connector.Kernel;
User user = objects[0] as User;
```

Si la cmdlet PowerShell retourne plusieurs objets, ils sont tous disponibles dans le tableau d'objets.

Selon les cas, l'appelant doit s'assurer que le tableau d'objets n'est pas vide et que les éléments ne sont pas « null ».

4.3.2 Deuxième surcharge de la méthode *Execute*

Cet exemple montre l'utilisation du tableau KeyValuePair pour chiffrer plusieurs fichiers :

```
string[] filePaths = new string[] { "file-path-1", "file-path-2", ... };
objects[] certificates; // retrieved from a call to Get-SDSCertificate API
KeyValuePair<string, object>[] parameters = new KeyValuePair<string,
object>[]
{
  new KeyValuePair<string, object>("-Path", filePaths),
  new KeyValuePair<string, object>("-Coworkers", certificates)
};
objects = api.Execute("Protect-SDSFile", parameters);
```

Chaque paramètre doit être ajouté au tableau d'objets KeyValuePair, la clé étant le nom du paramètre de la cmdlet.

4.4 Exécuter des exemples de code

Des projets d'exemples sont fournis avec Visual Studio 2013.

Prérequis :

- Microsoft .NET Framework 4.5.2 (https://www.microsoft.com/fr-fr/download/details.aspx?id=42643)
- Microsoft .NET Framework 4.5.2 Developer Pack (https://www.microsoft.com/fr-fr/download/details.aspx?id=42637)

Le .NET Framework 4.5.2 Developer Pack n'est nécessaire que pour compiler l'application utilisant Stormshield Data Connector. L'exécution d'une application utilisant Stormshield Data Connector ne le requiert pas.

Chaque projet fourni dans cette solution illustre l'utilisation d'une API donnée.

4.5 Cas d'usage des API .NET

Dans les exemples suivants, les cas d'erreur ne sont pas gérés.

4.5.1 Gérer la connexion utilisateur

Connecter un utilisateur

L'API Connect-SDUser permet de connecter un utilisateur à Stormshield Data Security.

```
using Stormshield.DataSecurity.Connector;
using Stormshield.DataSecurity.Connector.Kernel;
using (API api = new API())
{
  object[] objects = api.Execute("Connect-SDSUser <id> <password>");
  User connectedUser = objects[0] as User;
  Console.WriteLine("User '{0}' connected", connectedUser.Id);
}
```

Vérifier l'état de la connexion

L'API Get-SDSUser permet de connaître l'état de la session Stormshield Data Security courante (connecté ou verrouillé) :

```
using Stormshield.DataSecurity.Connector;
using Stormshield.DataSecurity.Connector.Kernel;
using (API api = new API())
{
  object[] objects = api.Execute("Get-SDSUser");
  User connectedUser = objects[0] as User;
  if (connectedUser.Locked)
    Console.WriteLine("User is locked");
}
```

Verrouiller une session

L'API Lock-SDSUser permet de verrouiller la session Stormshield Data Security courante :

```
using Stormshield.DataSecurity.Connector;
using (API api = new API())
{
  // first ensure that a user is connected
```

```
  api.Execute("Lock-SDSUser");
}
```

Déverrouiller une session

L'API Unlock-SDSUser permet de déverrouiller la session Stormshield Data Security courante :

```
using Stormshield.DataSecurity.Connector;
using Stormshield.DataSecurity.Connector.Kernel;
using (API api = new API())
{
  // first ensure that a user is connected and locked
  object[] objects = api.Execute("Unlock-SDSUser <password>");
  User connectedUser = objects[0] as User;
  Console.WriteLine("User '{0}' unlocked", connectedUser.Id);
}
```

Déconnecter un utilisateur

L'API Disconnect-SDSUser permet de déconnecter l'utilisateur courant :

```
using Stormshield.DataSecurity.Connector;
using (API api = new API())
{
  api.Execute("Disconnect-SDSUser");
  // if no exception thrown, then the operation succeeded
}
```

4.5.2 Interagir avec l'annuaire de confiance

Un utilisateur Stormshield Data Security doit être connecté pour pouvoir effectuer les opérations décrites ci-dessous (reportez-vous à Gérer la connexion utilisateur). Si aucun utilisateur n'est connecté, une exception est levée.

Récupérer les certificats

L'API Get-SDSCertificate permet de récupérer un ou plusieurs certificats présents dans l'annuaire.

```
using Stormshield.DataSecurity.Connector;
using Stormshield.DataSecurity.Connector.Common;
using (API api = new API())
{
  // first ensure that a user is connected
  object[] objects = api.Execute("Get-SDSCertificate -EmailAddress
alicesmith@mycompany.com");
  X509Certificate certificate = objects[0] as X509Certificate;
}
```

> **NOTE**
> La récupération des certificats par adresse e-mail se base uniquement sur le champ **Adresse email** disponible dans l'onglet *Général* d'un certificat. Les adresses e-mail contenues dans les détails du certificat ne sont pas utilisées.

Il est possible de récupérer plusieurs certificats en même temps en utilisant la syntaxe de liste PowerShell :

```
using Stormshield.DataSecurity.Connector;
using (API api = new API())
{
  // first ensure that a user is connected
  object[] objects = api.Execute("Get-SDSCertificate -EmailAddress
alicesmith@mycompany.com,jodiefisher@mycompany.com,robertmiller@mycompany.
com");
  X509Certificate aliceSmithCertificate = objects[0] as X509Certificate;
  X509Certificate jodieFisherCertificate = objects[1] as X509Certificate;
  X509Certificate robertMillerCertificate = objects[2] as X509Certificate;
}
```

Si l'option UpdateStatus n'est pas sélectionnée lors de la commande, les certificats sont récupérés sans la donnée de statut. Pour avoir une mise à jour de ce champ, il faut le demander explicitement. La commande prend alors plus de temps à renvoyer les certificats.

Exporter le contenu de l'annuaire

Le contenu de l'annuaire de confiance peut être sauvegardé dans un fichier pour être restauré ultérieurement.

L'API Export-SDSAddressBook permet d'exporter une partie du contenu de l'annuaire d'un utilisateur.

L'exemple suivant montre comment exporter dans un fichier au format P7B, les certificats, leur parenté ainsi que les contacts et les groupes :

```
using Stormshield.DataSecurity.Connector;
using (API api = new API())
{
  // first ensure that a user is connected
  string p7bPath = @"path\to\file.p7b";
  object[] objects = api.Execute(string.Format("Export-SDSAddressBook -
Path '{0}' -ExportAncestry -ExportContactsAndGroups", p7bPath));
  System.IO.FileInfo p7bFileInfo = objects[0] as System.IO.FileInfo;
}
```

L'export de la totalité de l'annuaire en incluant la personnalisation des certificats s'effectue en utilisant l'API Backup-SDSAddressBook.

L'exemple qui suit permet de sauvegarder la totalité de l'annuaire dans un fichier au format P7Z :

```
using Stormshield.DataSecurity.Connector;
using (API api = new API())
{
  // first ensure that a user is connected
  string p7zPath = @"path\to\file.p7z";
  oject[] objects = api.Execute(string.Format("Backup-SDSAddressBook -Path
'{0}'", p7zPath));
  System.IO.FileInfo p7zFileInfo = objects[0] as System.IO.FileInfo;
}
```

Le fichier *.p7z* permet de sauvegarder les données personnalisées des certificats. En revanche, ce format n'est pas compatible avec les versions de Stormshield Data Security antérieures à la version 9.1.

Importer le contenu de l'annuaire

Le contenu de l'annuaire peut être importé avec le code suivant. L'exemple présenté ici utilise un fichier au format P7Z, mais tout format compatible avec l'annuaire de la Suite pourrait fonctionner.

```
using Stormshield.DataSecurity.Connector;
using (API api = new API())
{
  // first ensure that a user is connected
  string p7zPath = @"path\to\file.p7z";
  api.Execute(string.Format("Import-SDSAddressBook -Path '{0}'",
p7zPath));
}
```

4.5.3 Interagir avec Stormshield Data File

Obtenir des informations sur des fichiers

L'API Get-SDSFile permet de récupérer des informations sur un fichier, chiffré ou non.

```
using Stormshield.DataSecurity.Connector;
using Stormshield.DataSecurity.Connector.File;
using (API api = new API())
{
  string sboxPath = @"path\to\file.sdsx";
  object[] objects = api.Execute(string.Format("Get-SDSFile '{0}'",
sboxPath));
  SecureFile secureFile = objects[0] as SecureFile;
}
```

L'objet SecureFile contient les informations suivantes sur le fichier :

- si le fichier est chiffré ou pas.
- la liste de destinataires (adresses e-mail, dans le cas où le fichier est chiffré). Information lue dans l'en-tête.

Plusieurs chemins d'accès de fichiers peuvent être donnés dans l'API Get-SDSFile, en tant que liste PowerShell :

```
object[] objects = api.Execute(string.Format("Get-SDSFile '{0}','{1}'",
sboxPath1, sboxPath2));
```

Dans ce cas, le tableau d'objets retourné contient deux éléments procurant les informations sur chaque fichier.

Il n'est pas nécessaire que l'utilisateur soit connecté pour récupérer les informations de base d'un fichier. Pour obtenir plus d'informations (concernant les certificats), un utilisateur doit être connecté.

Chiffrer un fichier

L'API Protect-SDSFile permet de chiffrer un fichier avec Stormshield Data File. La liste des certificats doit être récupérée de l'annuaire de l'utilisateur (reportez-vous à Récupérer les certificats).

Un utilisateur Stormshield Data Security doit être connecté pour chiffrer un fichier (reportez-vous à Gérer la connexion utilisateur). Si aucun utilisateur n'est connecté, une exception est levée.

L'exemple suivant montre comment chiffrer un fichier pour plusieurs collaborateurs en fournissant leur certificat :

```
using Stormshield.DataSecurity.Connector;
using Stormshield.DataSecurity.Connector.File;
using (API api = new API())
```

```
{
  // first ensure that a user is connected
  object[] certificates = api.Execute("Get-SDSCertificate -EmailAddress
alicesmith@mycompany.com,jodiefisher@mycompany.com");
  KeyValuePair<string, object>[] parameters = new KeyValuePair<string,
object>[]
  {
    new KeyValuePair<string, object>("-Path",
@"path\to\file\to\be\encrypted"),
    new KeyValuePair<string, object>("-Coworkers", certificates)
  };
  object[] objects = api.Execute("Protect-SDSFile", parameters);
  SecureFile secureFile = objects[0] as SecureFile;
}
```

Pour chiffrer plusieurs fichiers en même temps :

```
string[] files = new string[] { @"some\path", @"some\other\path" };
KeyValuePair<string, object>[] parameters = new KeyValuePair<string,
object>[]
{
  new KeyValuePair<string, object>("-Path", files),
  new KeyValuePair<string, object>("-Coworkers", objects)
};
```

Déchiffrer un fichier

L'API Unprotect-SDSFile permet de déchiffrer un fichier avec Stormshield Data File.

Un utilisateur Stormshield Data Security doit être connecté pour déchiffrer un fichier (reportez-vous à Gérer la connexion utilisateur). Si aucun utilisateur n'est connecté, une exception est générée.

```
using Stormshield.DataSecurity.Connector;
using Stormshield.DataSecurity.Connector.File;
using (API api = new API())
{
  // first ensure that a user is connected
  string sboxPath = @"path\to\file.sdsx";
  object[] objects = api.Execute(string.Format("Unprotect-SDSFile '{0}'",
filePath));
  SecureFile secureFile = objects[0] as SecureFile;
}
```

Ajouter des collaborateurs pour des fichiers sécurisés

L'API Add-SDSFileCoworker permet de rajouter un ou plusieurs collaborateurs sur un ou plusieurs fichiers chiffrés avec Stormshield Data File.

Un utilisateur Stormshield Data Security doit être connecté pour pourvoir ajouter des collaborateurs (reportez-vous à Gérer la connexion utilisateur). Si aucun utilisateur n'est connecté, une exception est générée.

```
using Stormshield.DataSecurity.Connector;
using Stormshield.DataSecurity.Connector.Common;
using Stormshield.DataSecurity.Connector.File;
using (API api = new API())
{
  // first ensure that a user is connected
  string sboxPath = @"path\to\file.sdsx";
  object[] certificates = api.Execute("Get-SDSCertificate -EmailAddress
alicesmith@mycompany.com");
```

```
  KeyValuePair<string, object>[] parameters = new KeyValuePair<string,
object>[]
  {
    new KeyValuePair<string, object>("Path", sboxPath),
    new KeyValuePair<string, object>("Coworkers", certificates)
  };
  object[] objects = api.Execute("Add-SDSFileCoworker", parameters);
  SecureFile secureFile = objects[0] as SecureFile;
}
```

Changer la liste des collaborateurs associés à des fichiers sécurisés

L'API Set-SDSFileCoworker permet de remplacer tous les collaborateurs associés à un ou plusieurs fichiers chiffrés avec Stormshield Data File par un ou plusieurs autres.

Un utilisateur Stormshield Data Security doit être connecté pour pourvoir ajouter des collaborateurs (reportez-vous à Gérer la connexion utilisateur). Si aucun utilisateur n'est connecté, une exception est générée.

```
using Stormshield.DataSecurity.Connector;
using Stormshield.DataSecurity.Connector.Common;
using Stormshield.DataSecurity.Connector.File;
using (API api = new API())
{
  // first ensure that a user is connected
  string sboxPath = @"path\to\file.sdsx";
  object[] certificates = api.Execute("Get-SDSCertificate -EmailAddress
alicesmith@mycompany.com");
  KeyValuePair<string, object>[] parameters = new KeyValuePair<string,
object>[]
  {
    new KeyValuePair<string, object>("Path", sboxPath),
    new KeyValuePair<string, object>("Coworkers", certificates)
  };
  object[] objects = api.Execute("Set-SDSFileCoworker", parameters);
  SecureFile secureFile = objects[0] as SecureFile;
}
```

Supprimer des collaborateurs associés à des fichiers sécurisés

L'API Remove-SDSFileCoworker permet de supprimer un ensemble de collaborateurs associés à un ou plusieurs fichiers chiffrés avec Stormshield Data File.

Un utilisateur Stormshield Data Security doit être connecté pour pourvoir ajouter des collaborateurs (reportez-vous à Gérer la connexion utilisateur). Si aucun utilisateur n'est connecté, une exception est générée.

```
using Stormshield.DataSecurity.Connector;
using Stormshield.DataSecurity.Connector.Common;
using Stormshield.DataSecurity.Connector.File;
using (API api = new API())
{
  // first ensure that a user is connected
  string sboxPath = @"path\to\file.sdsx";
  object[] objects = api.Execute(string.Format("Remove-SDSFileCoworker '
{0}' -EmailAddress alicesmith@mycompany.com", sboxPath));
  SecureFile secureFile = objects[0] as SecureFile;
}
```

Il est possible de supprimer des collaborateurs qui sont soit identifiés par leur adresse e-mail soit par leur certificat. Si la suppression se fait par rapport aux adresses e-mails, une exception sera levée si une des adresses d'un collaborateur est associée à un certificat manquant.

4.5.4 Interagir avec Stormshield Virtual Disk

Un utilisateur Stormshield Data Security doit être connecté pour pouvoir réaliser les opérations décrites ci-dessous (reportez-vous à Gérer la connexion utilisateur). Si aucun utilisateur n'est connecté, une exception est levée.

Créer un volume

L'API New-SDSDisk permet de créer un volume de disque virtuel chiffré.

L'exemple suivant montre comment créer un volume de 42 mégaoctets :

```
using Stormshield.DataSecurity.Connector;
using Stormshield.DataSecurity.Connector.VirtualDisk;
using (API api = new API())
{
  // first ensure that a user is connected
  string vboxPath = @"path\to\file.vbox";
  object[] objects = api.Execute(string.Format("New-SDSDisk '{0}' -Size
{1}", vboxPath, 50));
  Volume volume = objects[0] as Volume;
}
```

Dans cet exemple, le volume créé n'est ni formaté ni monté.

Obtenir des informations sur un volume

L'API Get-SDSDisk permet de récupérer des informations sur un volume de disque virtuel.

```
using Stormshield.DataSecurity.Connector;
using Stormshield.DataSecurity.Connector.VirtualDisk;
using (API api = new API())
{
  // first ensure that a user is connected
  string vboxPath = @"path\to\file.vbox";
  object[] objects = api.Execute(string.Format("Get-SDSDisk '{0}'",
vboxPath));
  Volume volume = obj as Volume;
}
```

Monter un volume

L'API Mount-SDSDisk permet de monter un volume de disque virtuel.

```
using Stormshield.DataSecurity.Connector;
using Stormshield.DataSecurity.Connector.VirtualDisk;
using (API api = new API())
{
  // first ensure that a user is connected
  string vboxPath = @"path\to\file.vbox";
  object[] objects = api.Execute(string.Format("Mount-SDSDisk '{0}'",
vboxPath));
  Volume volume = obj as Volume;
}
```

Démonter un volume

L'API Dismount-SDSDisk permet de démonter un volume de disque virtuel.

```
using Stormshield.DataSecurity.Connector;
using Stormshield.DataSecurity.Connector.VirtualDisk;
using (API api = new API())
```

```
{
  // first ensure that a user is connected
  string vboxPath = @"path\to\file.vbox";
  object[] objects = api.Execute(string.Format("Dismount-SDSDisk '{0}'",
vboxPath));
  Volume volume = obj as Volume;
}
```

Activer le montage automatique d'un volume

L'API Enable-SDSDiskAutomaticMount permet d'activer le montage automatique d'un volume de disque virtuel.

```
using Stormshield.DataSecurity.Connector;
using Stormshield.DataSecurity.Connector.VirtualDisk;
using (API api = new API())
{
  // first ensure that a user is connected
  string vboxPath = @"path\to\file.vbox";
  object[] objects = api.Execute(string.Format("Enable-
SDSDiskAutomaticMount '{0}'", vboxPath));
  Volume volume = obj as Volume;
}
```

Désactiver le montage automatique d'un volume

L'API Disable-SDSDiskAutomaticMount permet de désactiver le montage automatique d'un volume de disque virtuel.

```
using Stormshield.DataSecurity.Connector;
using Stormshield.DataSecurity.Connector.VirtualDisk;
using (API api = new API())
{
  // first ensure that a user is connected
  string vboxPath = @"path\to\file.vbox";
  object[] objects = api.Execute(string.Format("Disable-
SDSDiskAutomaticMount '{0}'", vboxPath));
  Volume volume = obj as Volume;
}
```

4.5.5 Interagir avec Stormshield Data Team

Créer une règle sur un dossier

Un utilisateur Stormshield Data Security doit être connecté pour créer une règle (reportez-vous à Gérer la connexion utilisateur). Si aucun utilisateur n'est connecté, une exception est levée.

La cmdlet New-SDSTeamRule permet de créer une règle sur un ou plusieurs dossiers avec Stormshield Data Team.

L'exemple suivant montre comment créer une règle pour plusieurs collaborateurs :

```
using Stormshield.DataSecurity.Connector;
using Stormshield.DataSecurity.Connector.Team;
using (API api = new API())
{
  // first ensure that a user is connected
  string folderPath = @"path\to\folder";
  object[] certificates = api.Execute("Get-SDSCertificate -EmailAddress
alicesmith@mycompany.com,jodiefisher@mycompany.com");
  KeyValuePair<string, object>[] parameters = new KeyValuePair<string,
```

```
object>[]
  {
    new KeyValuePair<string, object>("Path", folderPath),
    new KeyValuePair<string, object>("Coworkers", certificates)
  };
  object[] objects = api.Execute("New-SDSTeamRule", parameters);
  RuleInfoData ruleInfoData = objects[0] as RuleInfoData;
}
```

L'utilisateur connecté qui crée la règle est automatiquement renseigné en tant que propriétaire (paramètre Owners). Si aucun utilisateur n'est connecté alors la création de règle échoue.

Lire la règle associée à un dossier

Un utilisateur Stormshield Data Security doit être connecté pour lire une règle (reportez-vous à Gérer la connexion utilisateur). Si aucun utilisateur n'est connecté, une exception est levée.

La cmdlet Get-SDSTeamRule permet de lire la règle associée à un ou plusieurs dossiers avec Stormshield Data Team.

```
using Stormshield.DataSecurity.Connector;
using Stormshield.DataSecurity.Connector.Team;
using (API api = new API())
{
  // first ensure that a user is connected
  string folderPath = @"path\to\folder";
  object[] objects = api.Execute(string.Format("Get-SDSTeamRule '{0}'",
folderPath));
  RuleInfoData ruleInfoData = objects[0] as RuleInfoData;
}
```

Lire les informations d'un fichier

La cmdlet Get-SDSTeamFile permet de lire les informations Team d'un ou plusieurs fichiers avec Stormshield Data Team.

```
using Stormshield.DataSecurity.Connector;
using Stormshield.DataSecurity.Connector.Team;
using (API api = new API())
{
  // first ensure that a user is connected
  string filePath = @"path\to\file";
  object[] objects = api.Execute(string.Format("Get-SDSTeamFile '{0}'",
filePath));
  FileInfoData fileInfoData = objects[0] as FileInfoData;
}
```

Appliquer une règle sur un dossier

Un utilisateur Stormshield Data Security doit être connecté pour appliquer une règle (reportez-vous à Gérer la connexion utilisateur). Si aucun utilisateur n'est connecté, une exception est levée.

La cmdlet Protect-SDSTeam permet d'appliquer une règle sur un ou plusieurs dossiers avec Stormshield Data Team.

Cette règle doit avoir été créée au préalable.

```
using Stormshield.DataSecurity.Connector;
using Stormshield.DataSecurity.Connector.Team;
using (API api = new API())
{
```

```
  // first ensure that a user is connected
  string folderPath = @"path\to\folder";
  object[] objects = api.Execute(string.Format("Protect-SDSTeam '{0}',
folderPath));
  foreach (object o in objects)
  {
    OperationStatus status = o as OperationStatus;
  }
}
```

Supprimer une règle associée à un dossier

Un utilisateur Stormshield Data Security doit être connecté pour supprimer une règle (reportez-vous à Gérer la connexion utilisateur). Si aucun utilisateur n'est connecté, une exception est levée.

La cmdlet Remove-SDSTeamRule permet de supprimer une règle associée à un ou plusieurs dossiers avec Stormshield Data Team.

Cette règle doit avoir été créée au préalable.

```
using Stormshield.DataSecurity.Connector;
using (API api = new API())
{
  // first ensure that a user is connected
  string folderPath = @"path\to\folder";
  api.Execute(string.Format("Remove-SDSTeamRule '{0}'", folderPath));
}
```

Désécuriser un dossier ou un fichier protégé avec une règle

Un utilisateur Stormshield Data Security doit être connecté pour désécuriser un dossier ou un fichier (reportez-vous à Gérer la connexion utilisateur). Si aucun utilisateur n'est connecté, une exception est levée.

La cmdlet Unprotect-SDSTeam permet de désécuriser un ou plusieurs fichiers ou dossiers avec Stormshield Data Team.

La règle doit avoir été supprimée au préalable.

```
using Stormshield.DataSecurity.Connector;
using Stormshield.DataSecurity.Connector.Team;
using (API api = new API())
{
  // first ensure that a user is connected
  string folderPath = @"path\to\folder";
  object[] objects = api.Execute(string.Format("Unprotect-SDSTeam '{0}'",
folderPath));
  foreach (object o in objects)
  {
    OperationStatus status = o as OperationStatus;
  }
}
```

4.5.6 Utiliser les API .NET à partir d'un programme C++

Pour piloter Stormshield Data Security en utilisant les API .NET depuis un programme écrit en C ou en C++, vous devez mettre en place un pont entre les deux technologies en écrivant du code en C++ managé (« C++/CLI wrapper ») qui va permettre d'appeler du code .NET depuis du code C ou C++.

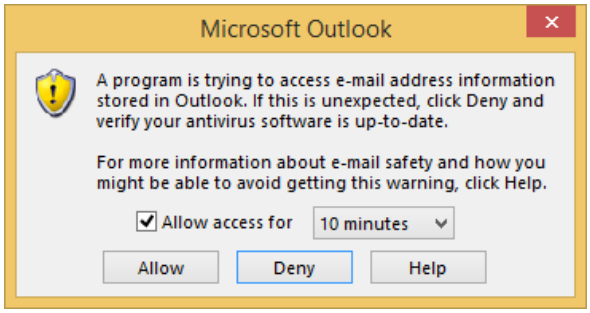
5. Piloter Stormshield Data Mail Édition Outlook

Stormshield Data Mail Édition Outlook peut être piloté par un script PowerShell ou un programme .NET pour envoyer et lire des messages chiffrés et/ou signés.

5.1 Configurer Microsoft Outlook

Microsoft Outlook restreint par défaut l'exécution de programmes externes.

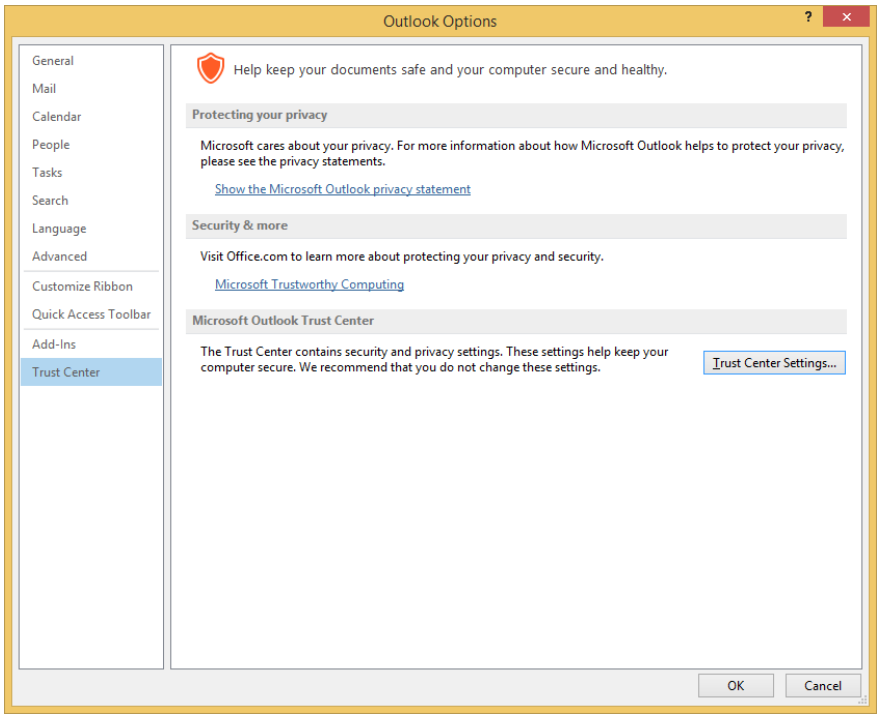
Lorsqu'un script PowerShell ou un programme .NET sont exécutés en utilisant les propriétés Microsoft Outlook, la fenêtre suivante s'ouvre :



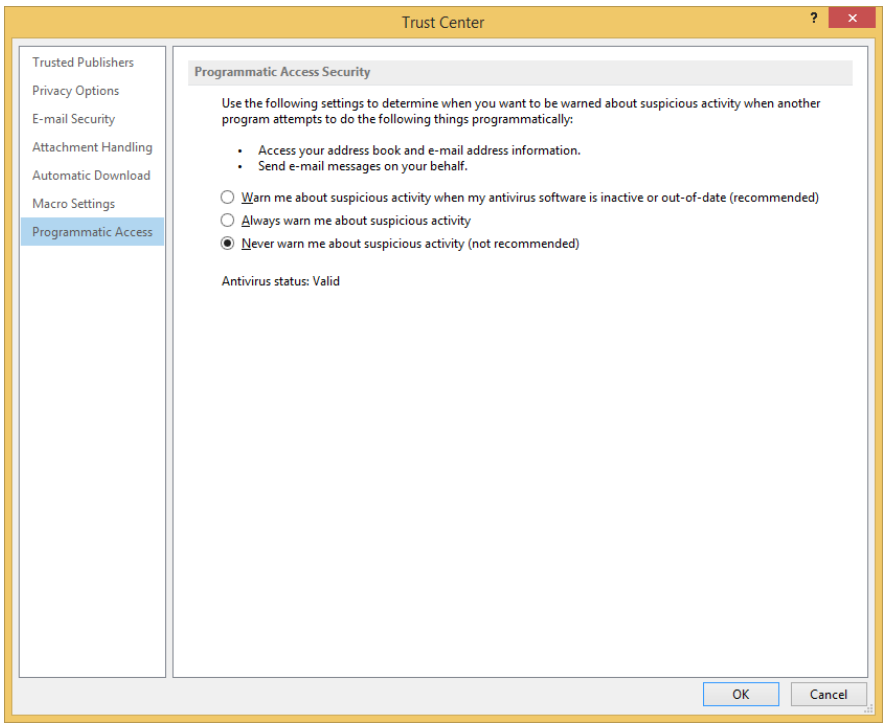
Cliquez sur **Accepter** pour autoriser le script PowerShell ou le programme .NET à piloter Microsoft Outlook. L'accès est autorisé pour une durée définie (1, 2, 5 ou 10 minutes).

Pour exécuter un script PowerShell ou un programme .NET via Microsoft Outlook et éviter cette fenêtre, configurez Microsoft Outlook afin de pouvoir l'utiliser à partir d'autres programmes :

1. Lancez Microsoft Outlook en tant qu'administrateur.
2. Ouvrez le Centre de gestion de la confidentialité dans les options Outlook et cliquez sur **Paramètres du Centre de gestion de la confidentialité** :



3. Sélectionnez **Ne jamais m'avertir des activités douteuses** dans l'onglet *Accès par programme*.



Vous pouvez configurer une clé de registre sur chaque poste de travail afin d'autoriser automatiquement l'exécution de programmes externes dans Microsoft Outlook. Pour Microsoft Outlook 2019 et Office 365, la clé de registre est :

- Microsoft Outlook 32 bits ou 64 bits :
  HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Office\16.0\Outlook\Security
  DWORD: ObjectModelGuard
  Value: 0, 1 or 2
- Microsoft Outlook 32 bits exécuté sur un système 64 bits :
  HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\Office\16.0\Outlook\Security
  DWORD: ObjectModelGuard
  Value: 0, 1 or 2

Les valeurs ObjectModelGuard dépendent des paramètres sélectionnés dans **Sécurité de l'accès par programme** :

- **M'avertir des activités douteuses lorsque mon logiciel antivirus est inactif ou n'est pas à jour (recommandé)** → Valeur **0**
- **Toujours m'avertir des activités douteuses** → Valeur **1**
- **Ne jamais m'avertir des activités douteuses (non recommandé)** → Valeur **2**

5.2 Configurer Stormshield Data Mail Édition Outlook

En mode de fonctionnement normal de l'add-in Stormshield Data Mail Édition Outlook, la valeur EnableScripting de la clé de registre **HKLM\SOFTWARE\Arkoon\Security BOX Enterprise\Properties\Mail** est désactivée (valeur = 0). Ainsi, durant l'envoi ou la lecture de messages chiffrés et/ou signés, si des erreurs surviennent (problème de certificat par exemple), des boîtes de dialogue nécessitant une réponse manuelle de l'utilisateur s'affichent.

Pour piloter l'add-in en mode "script", c'est-à-dire à partir d'un script PowerShell ou d'un programme .NET, il faut activer cette valeur de registre (valeur = 1). Ainsi les éventuelles erreurs ne provoqueront pas l'affichage de boîtes de dialogue nécessitant la réponse de l'utilisateur.

Si cette valeur n'est pas activée et que l'add-in est piloté en mode "script", les boîtes de dialogue qui pourraient s'ouvrir bloqueraient l'éxécution du processus d'envoi ou de lecture d'un message chiffré et/ou signé. Les éventuelles erreurs survenant durant le processus sont recensées dans le fichier de trace *%TEMP%\Arkoon.SecurityBox.Mail.Scripting.log*.

A la fin de l'exécution du processus d'envoi ou de lecture d'un message chiffré et/ou signé, il faut si nécessaire désactiver la valeur pour rebasculer sur un mode de fonctionnement normal de l'add-in Stormshield Data Mail Édition Outlook.

La valeur par défaut de la valeur de registre EnableScripting est 0.

5.3 Interagir avec Stormshield Data Mail Édition Outlook

Le module Stormshield Data Mail Édition Outlook traite les messages sortants pour les chiffrer et/ou les signer et les messages entrants pour les déchiffrer et/ou lire leur signature en fonction des propriétés affectées au message.

5.3.1 Envoyer un message chiffré et/ou signé

Créez un script PowerShell ou un programme .NET qui utiliseront l'interface de programmation *Outlook Object Model*.

Ajoutez les propriétés suivantes au message créé :

- **SDSEncrypt** pour chiffrer un message.
- **SDSSign** pour signer un message.

Le chiffrement est réalisé avec les certificats trouvés dans l'annuaire de l'utilisateur Stormshield Data Security courant. Si aucun utilisateur n'est connecté, une fenêtre de connexion s'ouvre.

La signature du message sera effectuée par l'utilisateur connecté à Stormshield Data Security, mettant en œuvre sa clé privée de signature.

Pour plus d'informations sur la création de nouvelles propriétés utilisateur dans la collection UserProperties et sur le comportement de sécurité de *Outlook Object Model*, consultez le site Web Microsoft Developer Network (MSDN).

Les exemples suivants illustrent les deux méthodes pour ouvrir une instance Microsoft Outlook et envoyer un message signé et chiffré.

Le script PowerShell et le programme . NET envoient automatiquement un message chiffré et signé au destinataire john.doe@mycompany.com. Ce message est également chiffré pour l'expéditeur.

PowerShell

```
Connect-SDSUser -Id "Alice Smith" -Interactive
Set-ItemProperty -Path "HKLM:\SOFTWARE\Arkoon\Security BOX
Enterprise\Properties\Mail" -Name "EnableScripting" -Value 1
$outlook = New-Object -ComObject Outlook.Application
$session = $outlook.Session
$session.Logon("Outlook") # maps to configured Outlook profile name
$mailItem = $outlook.CreateItem(0)
$mailItem.Recipients.Add("john.doe@mycompany.com")
```

```
$mailItem.Subject = "Simple text Subject"
$mailItem.Body = "Simple text Body"
$userProp = $mailItem.UserProperties.Add("SDSEncrypt", 6, $false, 1)
$userProp.Value = $true
$userProp = $mailItem.UserProperties.Add("SDSSign", 6, $false, 1)
$userProp.Value = $true
$mailItem.Send()
Set-ItemProperty -Path "HKLM:\SOFTWARE\Arkoon\Security BOX
Enterprise\Properties\Mail" -Name "EnableScripting" -Value 0
```

Le script ci-dessus ne tient pas compte de la libération des objets COM instanciés depuis Microsoft Outlook. Il est nécessaire de les libérer en utilisant :

```
[Runtime.InteropServices.Marshal]::ReleaseComObject($variable)
```

.NET

```
using Stormshield.DataSecurity.Connector;
using Microsoft.Office.Interop.Outlook;

namespace SDConnectorSample
{
  class Program
  {
    static void Main(string[] args)
    {
      string keyPath = @"SOFTWARE\ARKOON\Security BOX
Enterprise\Properties\Mail";

      #region Enabling registry key
      using (RegistryKey baseKey = RegistryKey.OpenBaseKey
(RegistryHive.LocalMachine, RegistryView.Registry64))
      {
        using (RegistryKey subKey = baseKey.OpenSubKey(keyPath, true))
        {
          subKey.SetValue("EnableScripting", 1);
        }
      }
      #endregion

      using (API api = new API())
      {
        api.Execute("Connect-SDSUser -Id 'Alice Smith' -Interactive");
      }

      Application outlook = new Application();
      NameSpace session = outlook.Session;
      session.Logon("Outlook"); // maps to configured Outlook profile name
      MailItem mailItem = outlook.CreateItem(OlItemType.olMailItem);
      mailItem.Recipients.Add("john.doe@mycompany.com");
      mailItem.Subject = "Simple text Subject";
      mailItem.Body = "Simple text Body";
      UserProperty userPropEncrypt = mailItem.UserProperties.Add
("SDSEncrypt", OlUserPropertyType.olYesNo, false,
OlFormatYesNo.olFormatYesNoYesNo);
      userPropEncrypt.Value = true;
      UserProperty userPropSign = mailItem.UserProperties.Add("SDSSign",
OlUserPropertyType.olYesNo, false, OlFormatYesNo.olFormatYesNoYesNo);
      userPropSign.Value = true;
      ((Microsoft.Office.Interop.Outlook._MailItem)mailItem).Send();

      #region Disabling registry key
      using (RegistryKey baseKey = RegistryKey.OpenBaseKey
```

```
(RegistryHive.LocalMachine, RegistryView.Registry64))
      {
        using (RegistryKey subKey = baseKey.OpenSubKey(keyPath, true))
        {
          subKey.SetValue("EnableScripting", 0);
        }
      }
      #endregion
    }
  }
}
```

Le programme ci-dessus ne tient pas compte de la libération des objets COM instanciés depuis Microsoft Outlook. Il est nécessaire de les libérer en utilisant :

```
Marshal.ReleaseComObject(variable)
```

5.3.2 Lire un message chiffré et/ou signé

Lors de la lecture d'un message depuis Microsoft Outlook par le biais d'un script PowerShell ou d'un programme .NET, l'add-in Stormshield Data Édition Outlook déchiffre le message de manière transparente. L'objet MailItem ainsi récupéré est en clair et deux propriétés permettent de déterminer si le message était chiffré et/ou signé :

- **SDSReadEncrypted** si le message est chiffré
- **SDSReadSigned** si le message est signé

Les exemples suivants illustrent les deux méthodes pour ouvrir une instance Microsoft Outlook et lire un message depuis la boîte de réception.

Powershell

```
Connect-SDSUser -Id "Alice Smith" -Interactive
Set-ItemProperty -Path "HKLM:\SOFTWARE\Arkoon\Security BOX
Enterprise\Properties\Mail" -Name "EnableScripting" -Value 1
$outlook = New-Object -ComObject Outlook.Application
$session = $outlook.Session
$session.Logon("Outlook") # maps to configured Outlook profile name
$mapi = $outlook.GetNamespace("MAPI")
$inbox = $mapi.GetDefaultFolder(6)
$items = $inbox.Items
$mailItem = $items.GetLast()
$userProp = $mailItem.UserProperties.Find("SDSReadEncrypted")
$isEncrypted = $userProp.Value
$userProp = $mailItem.UserProperties.Find("SDSReadSigned”)
$isSigned = $userProp.Value
Set-ItemProperty -Path "HKLM:\SOFTWARE\Arkoon\Security BOX
Enterprise\Properties\Mail" -Name "EnableScripting" -Value 0
```

Le script ci-dessus ne tient pas compte de la libération des objets COM instanciés depuis Microsoft Outlook. Il est nécessaire de les libérer en utilisant :

```
[Runtime.InteropServices.Marshal]::ReleaseComObject($variable)
```

.NET

```
using Stormshield.DataSecurity.Connector;
using Microsoft.Office.Interop.Outlook;
```

```
namespace SDConnectorSample
{
  class Program
  {
    static void Main(string[] args)
    {
      string keyPath = @"SOFTWARE\ARKOON\Security BOX
Enterprise\Properties\Mail";

      #region Activation de la clé de registre
      using (RegistryKey baseKey = RegistryKey.OpenBaseKey
(RegistryHive.LocalMachine, RegistryView.Registry64))
      {
        using (RegistryKey subKey = baseKey.OpenSubKey(keyPath, true))
        {
          subKey.SetValue("EnableScripting", 1);
        }
      }
      #endregion

      using (API api = new API())
      {
        api.Execute("Connect-SDSUser -Id 'Alice Smith' -Interactive");
      }

      Application outlook = new Application();
      NameSpace session = outlook.Session;
      session.Logon("Outlook"); // maps to configured Outlook profile name
      NameSpace mapi = outlook.GetNamespace("MAPI");
      MAPIFolder inbox = mapi.GetDefaultFolder
(OlDefaultFolders.olFolderInbox);
      Items items = inbox.Items;
      MailItem mailItem = items.GetLast();
      UserProperty userPropEncrypted = mailItem.UserProperties.Find
("SDSReadEncrypted");
      bool isEncrypted = userPropEncrypted.Value;
      UserProperty userPropSigned = mailItem.UserProperties.Find
("SDSReadSigned");
      bool isSigned = userPropSigned.Value;

      #region Désactivation de la clé de registre
      using (RegistryKey baseKey = RegistryKey.OpenBaseKey
(RegistryHive.LocalMachine, RegistryView.Registry64))
      {
        using (RegistryKey subKey = baseKey.OpenSubKey(keyPath, true))
        {
          subKey.SetValue("EnableScripting", 0);
        }
      }
      #endregion
    }
  }
}
```

Le programme ci-dessus ne tient pas compte de la libération des objets COM instanciés depuis Microsoft Outlook. Il est nécessaire de les libérer en utilisant :

```
Marshal.ReleaseComObject(variable)
```

6. Désactiver Stormshield Data Connector

Si le composant Stormshield Data Connector est installé mais que son utilisation n'est pas souhaitée, vous pouvez le désactiver.

Pour désactiver Stormshield Data Connector, vous pouvez entre autres supprimer la permission **Lecture et Exécution** du fichier suivant :

```
<dossier d'installation>\Connector\Modules
↳ Stormshield.DataSecurity.Connector.PowerShell\
↳ Stormshield.DataSecurity.Connector.PowerShell.dll
```

7. Limitation

Le composant Stormshield Data Connector n'est pas « Thread-Safe ». L'utilisateur des cmdlets/API doit donc bien veiller à protéger son code par des structures d'exclusion mutuelles (« Mutex ») si nécessaire.

Annexe A. Exemple de cas d'usage

Le même exemple de cas d'usage est présenté ici en script PowerShell et en programme .NET.

L'administrateur d'un parc de machines souhaite qu'un dossier donné soit en permanence chiffré sur les postes des utilisateurs.

Lorsqu'un utilisateur se connecte à sa session Windows, l'administrateur souhaite :

- Connecter l'utilisateur à son compte Stormshield Data Security (en mode interactif) ;
- S'assurer qu'un dossier donné est sécurisé avec le module Stormshield Data Team (et le créer si besoin) ;
- Informer l'utilisateur de ce qu'il s'est passé.

Le dossier doit être systématiquement sécurisé par une règle Team et les fichiers chiffrés.

Ce scénario peut être implémenté à l'identique dans les deux modes d'utilisation du module Stormshield Data Connector.

Dans les deux implémentations, le dossier configuré est un dossier nommé « Secured » sur le bureau de l'utilisateur.

L'administrateur peut exécuter le script PowerShell ou le programme .NET au démarrage de la session Windows de l'utilisateur.

A.1 PowerShell

```
$securedFolder = Join-Path ([Environment]::GetFolderPath('Desktop'))
'Secured'

[Reflection.Assembly]::LoadWithPartialName('System.Windows.Forms')

try
{
  Connect-SDSUser -Interactive

  $report = ''

  if (-not (Test-Path -Path "$securedFolder"))
  {
    New-Item -Path "$securedFolder" -Type Directory | Out-Null
    $report += ("Folder '$securedFolder' has been created." +
[Environment]::NewLine)
  }

  try
  {
    $rule = Get-SDSTeamRule -Path "$securedFolder"
  }
  catch [Stormshield.DataSecurity.Connector.Team.RuleNeedUpdateException]
  {
  }
  if ($rule -ne $null -and $rule.Secured -eq $false)
  {
    New-SDSTeamRule -Path "$securedFolder"
    $report += ("Rule has been created on folder '$securedFolder'." +
[Environment]::NewLine)
  }

  Protect-SDSTeam -Path "$securedFolder"
```

```
  $report += ("Folder '$securedFolder' has been protected." +
[Environment]::NewLine)

  [Windows.Forms.MessageBox]::Show($report)
}
catch
{
  [Windows.Forms.MessageBox]::Show($_.Exception)
}
```

A.2 .NET

```
using Stormshield.DataSecurity.Connector;
using Stormshield.DataSecurity.Connector.Common;
using Stormshield.DataSecurity.Connector.Kernel;
using Stormshield.DataSecurity.Connector.Team;

namespace SecuredFolder
{
  static class Program
  {
    private static string SecuredFolder = Path.Combine
(Environment.GetFolderPath(Environment.SpecialFolder.Desktop), "Secured");

    static void Main()
    {
      string value = ConfigurationManager.AppSettings["SecuredFolder"];
      if (!string.IsNullOrEmpty(value))
        SecuredFolder = value;

      try
      {
        using (Stormshield.DataSecurity.Connector.API api = new
Stormshield.DataSecurity.Connector.API())
        {
          api.Execute("Connect-SDSUser -Interactive");

          string report = string.Empty;

          if (!Directory.Exists(SecuredFolder))
          {
            Directory.CreateDirectory(SecuredFolder);
            report += string.Format("Folder '{0}' has been created.{1}",
SecuredFolder, Environment.NewLine);
          }

          object[] objects = null;
          try
          {
            objects = api.Execute(string.Format("Get-SDSTeamRule -Path '
{0}'", SecuredFolder));
          }
          catch
(Stormshield.DataSecurity.Connector.Team.RuleNeedUpdateException)
          {
          }
          if (objects != null && objects.Length == 1)
          {
            RuleInfoData rule = objects[0] as RuleInfoData;
            if (!rule.Secured)
            {
              api.Execute(string.Format("New-SDSTeamRule -Path '{0}'",
```

```
SecuredFolder));
                report += string.Format("Rule has been created on folder '
{0}'.{1}", SecuredFolder, Environment.NewLine);
              }
            }

            api.Execute(string.Format("Protect-SDSTeam -Path '{0}'",
SecuredFolder));
            report += string.Format("Folder '{0}' has been protected.{1}",
SecuredFolder, Environment.NewLine);

            MessageBox.Show(report);
          }
        }
        catch (System.Exception exception)
        {
          MessageBox.Show(exception.ToString());
        }
      }
    }
}
```

Annexe B. Liste des cmdlets Stormshield Data Connector

Cette annexe propose une description et des informations sur chaque cmdlet Stormshield Data Connector.

Ces informations techniques sont disponibles uniquement en anglais.

B.1 Add-SDSFileCoworker

Summary

Adds coworkers to one or more files encrypted with Stormshield Data File.

Description

The Add-SDSFileCoworker cmdlet adds one or more coworkers to the coworker list of files encrypted with Stormshield Data File. It invokes transciphering mechanisms.

Parameters

Name	Required	Position	Default value	Pipeline	Wildcard characters	Description
-Path <String[]>	true	1		true (ByPropertyName)	false	Specifies the path to one or more files encrypted with Stormshield Data File.
-Coworkers <X509Certificate[]>	true	2		true (ByPropertyName)	false	Specifies one or more X.509 certificates to add to the encrypted file. Certificates will be added as coworkers.

Inputs

System.String[], Stormshield.DataSecurity.Connector.Common.X509Certificate[]

You can pipe the list of files to be transciphered or the list of X.509 certificates to add.

Outputs

Stormshield.DataSecurity.Connector.File.SecureFile[]

This object represents an array of files transciphered by Stormshield Data File. The SecureFile represents a file encrypted with Stormshield Data File.

Notes

If no user is connected, an exception is raised. If a user is connected but locked, an exception is raised.

Examples

Add coworkers to an encrypted file

```
C:\PS>$certificates = Get-SDSCertificate -Name 'Jodie FISHER','Robert MILLER'
Add-SDSFileCoworker 'C:\My Folder\Document.docx.sdsx' -Coworkers $certificates
```

This command adds the coworkers Jodie Fisher and Robert Miller to the file 'C:\Document.docx.sdsx'.

```
Path                : C:\My Folder\Document.docx.sdsx
Encrypted           : True
OriginalFileName    : C:\My Folder\Document.docx
Size                : 159712
Compressed          : False
Executable          : False
Mechanism           : AES 256
Author              : Alice SMITH
Coworkers           : {alice.smith@mycompany.com, jodie.fisher@mycompany.com, robert.miller@mycompany.com}
Certificates        : {Alice SMITH, Jodie FISHER, Robert MILLER}
CertRetrievalStatus : SUCCEEDED
```

B.2 Backup-SDSAddressBook

Summary

Backups the user's address book into a .p7z file

Description

The Backup-SDSAddressBook backups the whole address book content, including personalized data, into a .p7z file that can be restored later.

Parameters

Name	Required	Position	Default value	Pipeline	Wildcard characters	Description
-Path <String>	true	1		true (ByValue)	false	Specifies the path to the output .p7z file.

Inputs

System.String

You can pipe a string containing the relative or absolute path of the output .p7z file.

Outputs

System.IO.FileInfo

The System.IO.FileInfo object represents the output .p7z file.

Notes

Examples

Backup the whole address book content

```
C:\PS>Backup-SDSAddressBook 'C:\My Folder\addressbook.p7z'
```

This command backups the whole address book content into a .p7z file.

```
Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
-a---         07/08/2015     10:46       8059 addressbook.p7z
```

B.3 Connect-SDSUser

Summary

Opens a Stormshield Data Security session.

Description

The Connect-SDSUser cmdlet connects a user to its Stormshield Data Security account.

Parameters

Name	Required	Position	Default value	Pipeline	Wildcard characters	Description
-Id <String>	false	1		false	false	Specifies the identifier of the user to be connected. Identifier or .usr file path are permitted. Note: The user identifier is not case sensitive. In non-interactive mode: This parameter is required. If omitted, a System.ArgumentNullException exception is raised. If the string is empty, a System.ArgumentException exception is raised. If the account does not exist, a UnknownUserException exception is raised. If a session is already open, a UserAlreadyConnectedException exception is raised. In interactive mode: This parameter is not required. If this parameter is omitted or if the string is empty, the connection window opens up and the identifier of the last successfully connected user is pre-filled. If no previous connection occurred, this field is empty. If you click the [Cancel] button in the connection window then a exception is raised and contains the E_LOGON_USER_CANCEL error code.

-Password <String>	false	2	Empty string	false	false	Specifies the password of the account. The password is the PIN of the smart card or USB token if applicable. Note: The password is case sensitive. If you enter your password incorrectly too many times (default is three tries), your account will be blocked. For example, with an account which three tries: First attempt, if the password is incorrect, a BadPasswordTwoAttemptsException exception is raised (two tries left). Second attempt, if the password is incorrect, a BadPasswordOneAttemptException exception is raised (one try left). Third attempt, if the password is incorrect, a BadPasswordAccountBlockedException exception is raised (account blocked). In non-interactive mode, this parameter is required. If omitted or if the string is empty, an exception is raised. (See the type of the exception above). In interactive mode, this parameter is optional. If omitted or if the string is empty, the connection window opens up with an empty password field. If [Cancel] button is clicked in the connection window, an exception is raised (with E_LOGON_USER_CANCEL error code).

-SecurePassword <SecureString>	false	named	false	false	Specifies the password of the account. The password is the PIN of the smart card or USB token if applicable. Note: The password is case sensitive. This parameter allows the password to be specified in a secured manner. If you enter your password incorrectly too many times (default is three tries), your account will be blocked. For example, with an account with three tries: First attempt, if the password is incorrect, a BadPasswordTwoAttemptsException exception is raised (two tries left). Second attempt, if the password is incorrect, a BadPasswordOneAttemptException exception is raised (one try left). Third attempt, if the password is incorrect, a BadPasswordAccountBlockedException exception is raised (account blocked). In non-interactive mode, this parameter is required. If omitted or if the string is empty, an exception is raised. (See the type of the exception above). In interactive mode, this parameter is optional. If omitted or if the string is empty, the connection window opens up with an empty password field. If [Cancel] button is clicked in the connection window, an exception is raised (with E_LOGON_USER_CANCEL error code). To generate the secure password, use the command `Read-Host "password" -AsSecureString | ConvertFrom-SecureString | Out-File C:\secured-password.pwd` where "password" must be replaced by the user password. For more information about the *ConvertFrom-SecureString* parameter, refer to Microsoft PowerShell help system. For more information about the encryption of the password, refer to Microsoft help about Windows Data Protection.
-Interactive <SwitchParameter>	false	named	false	false	Specifies that connection is to be made in interactive mode. The connection window opens up if the identifier or password are not fully specified. Otherwise a dialog box displays the connection progress.

Inputs

System.String, System.String, System.Security.SecureString, System.Management.Automation.SwitchParameter

Outputs

Stormshield.DataSecurity.Connector.Kernel.User

This object represents a Stormshield Data Security user account.

Notes

If a user is already connected, an exception is raised.

Examples

Connect a user to its Stormshield Data Security account

```
C:\PS>Connect-SDSUser alicesmith password
```

This command connects the user Alice Smith to its Stormshield Data Security account.

```
Id                    : alicesmith
Name                  : Alice Smith
Locked                : False
EmailAddresses        : {alice.smith@mycompany.com}
EncryptionCertificate : Alice Smith
SignatureCertificate  : Alice Smith
```

Connect a user to its Stormshield Data Security account

```
C:\PS>Connect-SDSUser 'C:\ProgramData\Arkoon\Security BOX\Users\alicesmith\alicesmith.usr' password
```

This command connects the user Alice Smith to its Stormshield Data Security account.

```
Id                    : alicesmith
Name                  : Alice Smith
Locked                : False
EmailAddresses        : {alice.smith@mycompany.com}
EncryptionCertificate : Alice Smith
SignatureCertificate  : Alice Smith
```

Connect a user to its Stormshield Data Security account

```
C:\PS>Read-Host "password" -AsSecureString | ConvertFrom-SecureString | Out-File C:\secured-password.pwd
$secureString = (Get-Content C:\secured-password.pwd | ConvertTo-SecureString)
Connect-SDSUser alicesmith -SecurePassword $secureString
```

This command connects the user Alice Smith to its Stormshield Data Security account. A object of type SecureString is used for specifying the password in a secured manner.

```
Id                    : alicesmith
Name                  : Alice Smith
Locked                : False
EmailAddresses        : {alice.smith@mycompany.com}
EncryptionCertificate : Alice Smith
SignatureCertificate  : Alice Smith
```

Connect last successfully connected user to Stormshield Data Security, displaying the connection window

```
C:\PS>Connect-SDSUser -Interactive
```

This command requests connection to Stormshield Data Security, displaying the connection window. The user identifier is pre-filled with the last successfully connected user.

```
Id                    : alicesmith
Name                  : Alice Smith
Locked                : False
EmailAddresses        : {alice.smith@mycompany.com}
EncryptionCertificate : Alice Smith
SignatureCertificate  : Alice Smith
```

Connect a user to Stormshield Data Security, displaying the connection window

```
C:\PS>Connect-SDSUser alicesmith -Interactive
```

This command requests connection to Stormshield Data Security, displaying the connection window. The user identifier is pre-filled with the identifier "alicesmith".

```
Id                    : alicesmith
Name                  : Alice Smith
Locked                : False
EmailAddresses        : {alice.smith@mycompany.com}
EncryptionCertificate : Alice Smith
SignatureCertificate  : Alice Smith
```

B.4 Disable-SDSDiskAutomaticMount

Summary

Configures a Virtual Disk volume to be mounted manually.

Description

The Disable-SDSDiskAutomaticMount cmdlet configures a Virtual Disk volume to be mounted manually.

Parameters

Name	Required	Position	Default value	Pipeline	Wildcard characters	Description
-Volume <Volume[]>	true	named		true (ByValue)	false	Specifies one or more Virtual Disk volume objects to be mounted manually.
-Path <String[]>	true	1		true (ByPropertyName)	false	Specifies one or more path to .vbox files to be mounted manually.
-Drive <String>	true	named		true (ByValue)	false	Specifies the drive unit of the Virtual Disk volume to be mounted manually. The drive unit must be specified in uppercase.

Inputs

System.String[], Stormshield.DataSecurity.Connector.VirtualDisk.Volume[], System.String

You can pipe an array of strings containing one or more path to .vbox files, an array of Virtual Disk volume objects or the drive unit of a Virtual Disk volume.

Outputs

Stormshield.DataSecurity.Connector.VirtualDisk.Volume

This object represents a Virtual Disk volume.

Notes

If no user is connected, an exception is raised. If a user is connected but locked, an exception is raised. If the Virtual Disk volume is already configured to be mounted manually, an exception is raised.

Examples

Configure a Virtual Disk volume to be mounted manually

```
C:\PS>Disable-SDSDiskAutomaticMount 'C:\My Folder\virtualdisk.vbox'
```

This command configures the specified Virtual Disk volume to be mounted manually. The Path parameter is used.

```
FullName       : C:\My Folder\virtualdisk.vbox
Size           : 15
Mechanism      : AES 256
Mounted        : True
MountLetter    : Z
AutomaticMount : False
AccessMode     : ReadWrite
Label          : Sample-Virtual-Disk
FileSystem     : eFSTYPE_NONE
Locked         : False
```

Configure a Virtual Disk volume to be mounted manually

```
C:\PS>$volume = Get-SDSDisk 'C:\My Folder\virtualdisk.vbox'
Disable-SDSDiskAutomaticMount -Volume $volume
```

This command configures the specified Virtual Disk volume to be mounted manually. The Volume parameter is used.

```
FullName        : C:\My Folder\virtualdisk.vbox
Size            : 15
Mechanism       : AES 256
Mounted         : True
MountLetter     : Z
AutomaticMount  : False
AccessMode      : ReadWrite
Label           : Sample-Virtual-Disk
FileSystem      : eFSTYPE_NONE
Locked          : False
```

Configure a Virtual Disk volume to be mounted manually

```
C:\PS>Mount-SDSDisk 'C:\My Folder\virtualdisk.vbox' -MountLetter Z
Disable-SDSDiskAutomaticMount -Drive Z
```

This command configures the specified Virtual Disk volume to be mounted manually. The Drive parameter is used.

```
FullName        : C:\My Folder\virtualdisk.vbox
Size            : 15
Mechanism       : AES 256
Mounted         : True
MountLetter     : Z
AutomaticMount  : False
AccessMode      : ReadWrite
Label           : Sample-Virtual-Disk
FileSystem      : eFSTYPE_NONE
Locked          : False
```

Configure two Virtual Disk volumes to be mounted manually

```
C:\PS>Disable-SDSDiskAutomaticMount 'C:\My Folder\virtualdisk1.vbox','C:\My Folder\virtualdisk2.vbox'
```

This command configures the two specified Virtual Disk volumes to be mounted manually. The Path parameter is used.

```
FullName        : C:\My Folder\virtualdisk1.vbox
Size            : 15
Mechanism       : AES 256
Mounted         : True
MountLetter     : Z
AutomaticMount  : False
AccessMode      : ReadWrite
Label           : Sample-Virtual-Disk-1
FileSystem      : eFSTYPE_NONE
Locked          : False

FullName        : C:\My Folder\virtualdisk2.vbox
Size            : 15
Mechanism       : AES 256
Mounted         : True
MountLetter     : Y
AutomaticMount  : False
AccessMode      : ReadWrite
Label           : Sample-Virtual-Disk-2
FileSystem      : eFSTYPE_NONE
Locked          : False
```

Configure two Virtual Disk volumes to be mounted manually

```
C:\PS>$volume1 = Get-SDSDisk 'C:\My Folder\virtualdisk1.vbox'
$volume2 = Get-SDSDisk 'C:\My Folder\virtualdisk2.vbox'
Disable-SDSDiskAutomaticMount -Volume $volume1,$volume2
```

This command configures the two specified Virtual Disk volumes to be mounted manually. The Volume parameter is used.

```
FullName        : C:\My Folder\virtualdisk1.vbox
Size            : 15
```

```
Mechanism      : AES 256
Mounted        : True
MountLetter    : Z
AutomaticMount : False
AccessMode     : ReadWrite
Label          : Sample-Virtual-Disk-1
FileSystem     : eFSTYPE_NONE
Locked         : False

FullName       : C:\My Folder\virtualdisk2.vbox
Size           : 15
Mechanism      : AES 256
Mounted        : True
MountLetter    : Y
AutomaticMount : False
AccessMode     : ReadWrite
Label          : Sample-Virtual-Disk-2
FileSystem     : eFSTYPE_NONE
Locked         : False
```

B.5 Disconnect-SDSUser

Summary

Closes a Stormshield Data Security session.

Description

The Disconnect-SDSUser cmdlet disconnects a user from its Stormshield Data Security account.

Parameters

Name	Required	Position	Default value	Pipeline	Wildcard characters	Description

Inputs

Outputs

void

Returns nothing.

Notes

If no user is connected, an exception is raised.

Examples

Disconnect currently connected user

```
C:\PS>Disconnect-SDSUser
```

This command disconnects the currently connected user from its Stormshield Data Security account.

B.6 Dismount-SDSDisk

Summary

Dismounts a Virtual Disk volume.

Description

The Dismount-SDSDisk dismounts a Virtual Disk volume.

Parameters

Name	Required	Position	Default value	Pipeline	Wildcard characters	Description
-Volume <Volume[]>	true	named		true (ByValue)	false	Specifies the Virtual Disk volume to be dismounted.
-Force <SwitchParameter>	false	named		false	false	Specifies that the Virtual Disk volume dismount is forced if it is in use.
-Path <String[]>	true	1		true (ByPropertyName)	false	Specifies the path to .vbox file to be dismounted.

Inputs

System.String[], Stormshield.DataSecurity.Connector.VirtualDisk.Volume[], System.Management.Automation.SwitchParameter

You can pipe an array of strings containing one or more path to .vbox files, an array of Virtual Disk volume objects or a flag to force dismount.

Outputs

Stormshield.DataSecurity.Connector.VirtualDisk.Volume

This object represents a Virtual Disk volume.

Notes

If no user is connected, an exception is raised. If a user is connected but locked, an exception is raised. If the Virtual Disk volume does not exist, an exception is raised.

Examples

Dismount a Virtual Disk volume

```
C:\PS>Dismount-SDSDisk 'C:\My Folder\virtualdisk.vbox'
```

This command dismounts the specified Virtual Disk volume. The Path parameter is used.

```
FullName        : C:\My Folder\virtualdisk.vbox
Size            : 15
Mechanism       : AES 256
Mounted         : False
MountLetter     : ?
AutomaticMount  : False
AccessMode      : Unspecified
Label           : Sample-Virtual-Disk
FileSystem      : eFSTYPE_NONE
Locked          : False
```

Dismount a Virtual Disk volume

```
C:\PS>$volume = Mount-SDSDisk 'C:\My Folder\virtualdisk.vbox'
Dismount-SDSDisk -Volume $volume
```

This command dismounts the specified Virtual Disk volume. The Volume parameter is used.

```
FullName        : C:\My Folder\virtualdisk.vbox
Size            : 15
Mechanism       : AES 256
Mounted         : False
MountLetter     : ?
AutomaticMount  : False
AccessMode      : Unspecified
Label           : Sample-Virtual-Disk
FileSystem      : eFSTYPE_NONE
Locked          : False
```

Force a Virtual Disk volume to be dismounted

```
C:\PS>$volume = Mount-SDSDisk 'C:\My Folder\virtualdisk.vbox'
Dismount-SDSDisk -Volume $volume -Force
```

This command dismounts the specified Virtual Disk volume. As the volume is in use, the Force parameter is specified.

```
FullName        : C:\My Folder\virtualdisk.vbox
Size            : 15
Mechanism       : AES 256
Mounted         : False
MountLetter     : ?
AutomaticMount  : False
AccessMode      : Unspecified
Label           : Sample-Virtual-Disk
```

```
FileSystem      : eFSTYPE_NONE
Locked          : False
```

Dismount two Virtual Disk volumes

```
C:\PS>Dismount-SDSDisk 'C:\My Folder\virtualdisk1.vbox','C:\My Folder\virtualdisk2.vbox'
```

This command dismounts the two specified Virtual Disk volumes. The Path parameter is used.

```
FullName        : C:\My Folder\virtualdisk1.vbox
Size            : 42
Mechanism       : AES 256
Mounted         : False
MountLetter     : ?
AutomaticMount  : False
AccessMode      : Unspecified
Label           : Sample-Virtual-Disk-1
FileSystem      : eFSTYPE_NTFS
Locked          : False

FullName        : C:\My Folder\virtualdisk2.vbox
Size            : 42
Mechanism       : AES 256
Mounted         : False
MountLetter     : ?
AutomaticMount  : False
AccessMode      : Unspecified
Label           : Sample-Virtual-Disk-2
FileSystem      : eFSTYPE_NTFS
Locked          : False
```

Dismount two Virtual Disk volumes

```
C:\PS>$volume1 = Get-SDSDisk 'C:\My Folder\virtualdisk1.vbox'
$volume2 = Get-SDSDisk 'C:\My Folder\virtualdisk2.vbox'
Dismount-SDSDisk -Volume $volume1,$volume2
```

This command dismounts the two specified Virtual Disk volumes. The Volume parameter is used.

```
FullName        : C:\My Folder\virtualdisk1.vbox
Size            : 42
Mechanism       : AES 256
Mounted         : False
MountLetter     : ?
AutomaticMount  : False
AccessMode      : Unspecified
Label           : Sample-Virtual-Disk-1
FileSystem      : eFSTYPE_NTFS
Locked          : False

FullName        : C:\My Folder\virtualdisk2.vbox
Size            : 42
Mechanism       : AES 256
Mounted         : False
MountLetter     : ?
AutomaticMount  : False
AccessMode      : Unspecified
Label           : Sample-Virtual-Disk-2
FileSystem      : eFSTYPE_NTFS
Locked          : False
```

B.7 Enable-SDSDiskAutomaticMount

Summary

Configures a Virtual Disk volume to be mounted automatically.

Description

The Disable-SDSDiskAutomaticMount cmdlet configures a Virtual Disk volume to be mounted automatically.

Parameters

Name	Required	Position	Default value	Pipeline	Wildcard characters	Description
-Volume <Volume>	true	named		true (ByValue)	false	Specifies the Virtual Disk volume object to be mounted automatically.
-MountLetter <String>	false	named		false	false	Specifies drive unit mount letter to be used when automatically mounting the Virtual Disk volume.
-Force <SwitchParameter>	false	named		false	false	Specifies that the Virtual Disk volume dismount is forced if it is in use. The configuration of an already mounted Virtual Disk volume to be mounted automatically requires a preliminary dismount.
-Path <String>	true	1		true (ByPropertyName)	false	Specifies the path to .vbox file to be mounted automatically.

-Drive <String>	true	named	true (ByValue)	false	Specifies the drive unit of the Virtual Disk volume to be mounted automatically. The drive unit must be specified in uppercase.

Inputs

System.String, Stormshield.DataSecurity.Connector.VirtualDisk.Volume, System.String, System.String,System.Management.Automation.SwitchParameter

You can pipe an array of strings containing one or more path to .vbox files, a Virtual Disk volume object, the drive unit of a Virtual Disk volume to be configured, the drive unit mount letter or a flag to force dismount.

Outputs

Stormshield.DataSecurity.Connector.VirtualDisk.Volume

This object represents a Virtual Disk volume.

Notes

If no user is connected, an exception is raised. If a user is connected but locked, an exception is raised. If the Virtual Disk volume is already configured to be mounted automatically, an exception is raised.

Examples

Configure a Virtual Disk volume to be mounted automatically

```
C:\PS>Enable-SDSDiskAutomaticMount 'C:\My Folder\virtualdisk.vbox'
```

This command configures the specified Virtual Disk volume to be mounted automatically. The Path parameter is used.

```
FullName        : C:\My Folder\virtualdisk.vbox
Size            : 15
Mechanism       : AES 256
Mounted         : True
MountLetter     : Z
AutomaticMount  : True
AccessMode      : ReadWrite
Label           : Sample-Virtual-Disk
FileSystem      : eFSTYPE_NONE
Locked          : False
```

Configure a Virtual Disk volume to be mounted automatically

```
C:\PS>Enable-SDSDiskAutomaticMount 'C:\My Folder\virtualdisk.vbox' -MountLetter Z
```

This command configures the specified Virtual Disk volume to be mounted automatically. The Path parameter is used and the drive unit mount letter is explicitly specified.

```
FullName        : C:\My Folder\virtualdisk.vbox
Size            : 15
Mechanism       : AES 256
Mounted         : True
MountLetter     : Z
AutomaticMount  : True
AccessMode      : ReadWrite
Label           : Sample-Virtual-Disk
FileSystem      : eFSTYPE_NONE
Locked          : False
```

Configure a Virtual Disk volume to be mounted automatically

```
C:\PS>$volume = Get-SDSDisk 'C:\My Folder\virtualdisk.vbox'
Enable-SDSDiskAutomaticMount -Volume $volume
```

This command configures the specified Virtual Disk volume to be mounted automatically. The Volume parameter is used.

```
FullName       : C:\My Folder\virtualdisk.vbox
Size           : 15
Mechanism      : AES 256
Mounted        : True
MountLetter    : Z
AutomaticMount : True
AccessMode     : ReadWrite
Label          : Sample-Virtual-Disk
FileSystem     : eFSTYPE_NONE
Locked         : False
```

Force a Virtual Disk volume to be mounted automatically

```
C:\PS>Mount-SDSDisk 'C:\My Folder\virtualdisk.vbox' -MountLetter Z
Enable-SDSDiskAutomaticMount -Drive Z -Force
```

This command configures the specified Virtual Disk volume to be mounted automatically on drive unit mount letter Z. As the volume is in use, the Force parameter is specified.

```
FullName       : C:\My Folder\virtualdisk.vbox
Size           : 15
Mechanism      : AES 256
Mounted        : True
MountLetter    : Z
AutomaticMount : True
```

```
AccessMode      : ReadWrite
Label           : Sample-Virtual-Disk
FileSystem      : eFSTYPE_NONE
Locked          : False
```

Configure two Virtual Disk volumes to be mounted automatically

```
C:\PS>Enable-SDSDiskAutomaticMount 'C:\My Folder\virtualdisk1.vbox','C:\My Folder\virtualdisk2.vbox'
```

This command configures the two specified Virtual Disk volumes to be mounted automatically. The Path parameter is used.

```
FullName        : C:\My Folder\virtualdisk1.vbox
Size            : 15
Mechanism       : AES 256
Mounted         : True
MountLetter     : Z
AutomaticMount  : True
AccessMode      : ReadWrite
Label           : Sample-Virtual-Disk-1
FileSystem      : eFSTYPE_NONE
Locked          : False

FullName        : C:\My Folder\virtualdisk2.vbox
Size            : 15
Mechanism       : AES 256
Mounted         : True
MountLetter     : Y
AutomaticMount  : True
AccessMode      : ReadWrite
Label           : Sample-Virtual-Disk-2
FileSystem      : eFSTYPE_NONE
Locked          : False
```

Configure two Virtual Disk volumes to be mounted automatically

```
C:\PS>$volume1 = Get-SDSDisk 'C:\My Folder\virtualdisk1.vbox'
$volume2 = Get-SDSDisk 'C:\My Folder\virtualdisk2.vbox'
Enable-SDSDiskAutomaticMount -Volume $volume1,$volume2
```

This command configures the two specified Virtual Disk volumes to be mounted automatically. The Volume parameter is used.

```
FullName        : C:\My Folder\virtualdisk1.vbox
Size            : 15
Mechanism       : AES 256
Mounted         : True
MountLetter     : Z
AutomaticMount  : True
AccessMode      : ReadWrite
Label           : Sample-Virtual-Disk-1
FileSystem      : eFSTYPE_NONE
Locked          : False

FullName        : C:\My Folder\virtualdisk2.vbox
Size            : 15
Mechanism       : AES 256
Mounted         : True
MountLetter     : Y
AutomaticMount  : True
AccessMode      : ReadWrite
Label           : Sample-Virtual-Disk-2
FileSystem      : eFSTYPE_NONE
Locked          : False
```

B.8 Export-SDSAddressBook

Summary

Backups the user's address book into a .p7b file

Description

The Export-SDSAddressBook exports all the certificates contained in user's address book. The address book can be exported with groups and certificates trust chain. Personalized data is not exported.

Parameters

Name	Required	Position	Default value	Pipeline	Wildcard characters	Description
-Path <String>	true	1		true (ByValue)	false	Specifies the path to the output .p7b file.
-ExportAncestry <SwitchParameter>	false	named		false	false	Specifies that certificates trust chain is to be included.
-ExportContactsAndGroups <SwitchParameter>	false	named		false	false	Specifies that contacts and groups of certificates are to be exported.

Inputs

System.String, System.Management.Automation.SwitchParameter, System.Management.Automation.SwitchParameter

You can pipe a string containing the relative or absolute path of the output .p7b file, a flag to include trust chain or a flag to include contacts and groups.

Outputs

System.IO.FileInfo

The System.IO.FileInfo object represents the output .p7b file.

Notes

Examples

Export only certificates

```
C:\PS>Export-SDSAddressBook C:\addressbook.p7b
```

This command exports all certificates of currently connected user's address book, excluding trust chain, contacts and groups.

```
Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
-a---         07/08/2015     10:46       8059 addressbook.p7b
```

Export all certificates including trust chain

```
C:\PS>Export-SDSAddressBook C:\addressbook.p7b -ExportAncestry
```

This commande exports all certificates of currently connected user's address book, including trust chain and excluding contacts and groups.

```
Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
-a---         07/08/2015     10:46       8059 addressbook.p7b
```

Export all certificates including contacts and groups

```
C:\PS>Export-SDSAddressBook C:\addressbook.p7b -ExportContactsAndGroups
```

```
Exports all certificates of currently connected user's address book, including contacts and groups.
```

```
Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
-a---         07/08/2015     10:46       8059 addressbook.p7b
```

B.9 Get-SDSCertificate

Summary

Retrieves a certificate from the address book of the currently connected user.

Description

This cmdlet retrieves a certificate or a group of coworkers certificates from the address book of the currently connected user.

Parameters

Name	Required	Position	Default value	Pipeline	Wildcard characters	Description
-EmailAddress <String[]>	false	named		false	false	Specifies one or more e-mail addresses corresponding to a certificate in the currently connected user's address book. Note: this parameter is not case sensitive. Limitation: for a successful retrieval with EmailAddress parameter, the address should be defined as a main e-mail address in the certificate description. A certificate will not be found if the address is only listed as an alternative identity.
-Name <String[]>	false	named		false	false	Specifies one or more certificate's subject common names in the currently connected user's address book. Note: this parameter is case sensitive.
-Group <String[]>	false	named		false	false	Specifies one or more group names in the currently connected user's address book. All certificates in specified groups are retrieved. Note: this parameter is case sensitive.
-Usage <CertificateKeyUsages>	false	named		true (ByValue)	false	Specifies the key usage of certificates to retrieve. This parameter is optional and can one or more of the following values: None, DecipherOnly, EncipherOnly, CRLSign, CertificateSign, KeyAgreement, DataEncipherment, KeyEncipherment, NonRepudiation and DigitalSignature. The default value is DataEncipherment and KeyEncipherment.
-UpdateStatus <SwitchParameter>	false	named		false	false	Specifies that the certificate's status needs to be computed. If this parameters is not specified, Status member of returned object is set to Unknown. Specifying this parameter involves certificates retrieval to be longer.

Inputs

System.String[], System.String[], System.String[], Stormshield.DataSecurity.Connector.Common.CertificateKeyUsages, SwitchParameter

You can pipe a string containing the coworker's name, a group name, an e-mail address, the usage of a certificate or a flag to force computing status.

Outputs

Stormshield.DataSecurity.Connector.Common.X509Certificate

This object represents the certificate retrieved from the address book.

Notes

If no user is connected, an exception is raised. If a user is connected but locked, an exception is raised. If none of the Name, Group or EmailAddress parameters are given, all the certificates present in the user's address book are retrieved, according to the Usage parameter specified. Limitation: this cmdlet does not return the authority certificates nor the certificate of the currently connected user's key-holder.

Examples

Retrieve certificate corresponding to an email address

```
C:\PS>Get-SDSCertificate -EmailAddress alicesmith@mycompany.com
```

This command retrieves the certificate of the specified user, based on its e-mail address. The status of the returned certificate is Unknown.

```
Data            : {48, 130, 2, 225...}
KeyUsages       : DataEncipherment, KeyEncipherment
Issuer          : MyCompany CA
Subject         : Alice Smith
```

```
EmailAddress   : alicesmith@mycompany.com
Version        : V3
StatusGeneral  : Unknown
StatusFull     : Unknown
EffectiveDate  : 19/07/2015 12:58:01
ExpirationDate : 19/07/2017 12:58:01
```

Retrieve certificate corresponding to an email address and a common name

```
C:\PS>Get-SDSCertificate -EmailAddress alicesmith@mycompany.com -Name 'Jodie FISHER' -UpdateStatus
```

This command retrieves two certificates. The first one owned by the user Alice Smith, based on her e-mail address, the second one owned by the user Jodie Fisher, based on her name. For each certificate, the status is updated.

```
Data           : {48, 130, 2, 219...}
KeyUsages      : DataEncipherment, KeyEncipherment
Issuer         : MyCompany CA
Subject        : Alice Smith
EmailAddress   : alicesmith@mycompany.com
Version        : V3
StatusGeneral  : Ok
StatusFull     : Ok
EffectiveDate  : 19/07/2015 12:58:01
ExpirationDate : 19/07/2017 12:58:01

Data           : {48, 130, 2, 211...}
KeyUsages      : DataEncipherment, KeyEncipherment
Issuer         : MyCompany CA
Subject        : Jodie Fisher
EmailAddress   : jodiefisher@mycompany.com
Version        : V3
StatusGeneral  : Ok
StatusFull     : Ok
```

```
EffectiveDate  : 10/09/2015 14:30:01
ExpirationDate : 10/09/2017 14:30:01
```

B.10 Get-SDSDisk

Summary

Retrieves information about one or more Stormshield Data Virtual Disk volumes.

Description

The Get-SDSDisk cmdlet retrieves information about one or more Stormshield Data Virtual Disk volumes.

Parameters

Name	Required	Position	Default value	Pipeline	Wildcard characters	Description
-Path <String[]>	true	1		true (ByPropertyName)	false	Specifies the path to .vbox file of the Virtual Disk volume.
-Drive <Char[]>	false	named		true (ByPropertyName)	false	Specifies the drive unit of the Virtual Disk volume.

Inputs

System.String[], System.Char[]

You can pipe an array of strings containing one or more .vbox paths or an array of chars containing one or more drive unit mounting letters.

Outputs

Stormshield.DataSecurity.Connector.VirtualDisk.Volume

This object represents a Virtual Disk volume.

Notes

If no parameter is provided, information about all currently mounted Virtual Disk volumes is returned. If no user is connected, an exception is raised.

Examples

Retrieve information about a Virtual Disk volume

```
C:\PS>Get-SDSDisk 'C:\My Folder\virtualdisk.vbox'
```

This command retrieves the specified Virtual Disk volume information. The Path parameter is used.

```
FullName       : C:\My Folder\virtualdisk.vbox
Size           : 12
Mechanism      : AES 256
Mounted        : False
MountLetter    : Z
AutomaticMount : False
AccessMode     : Unspecified
Label          : Sample-Virtual-Disk
FileSystem     : eFSTYPE_NTFS
Locked         : False
```

Retrieve information about several Virtual Disk volumes

```
C:\PS>Get-SDSDisk 'C:\My Folder\virtualdisk1.vbox','C:\My Folder\virtualdisk2.vbox'
```

This command retrieves the specified Virtual Disk volumes information. The Path parameter is used.

```
FullName        : C:\My Folder\virtualdisk1.vbox
Size            : 12
Mechanism       : AES 256
Mounted         : False
MountLetter     : Z
AutomaticMount  : False
AccessMode      : Unspecified
Label           : Sample-Virtual-Disk-1
FileSystem      : eFSTYPE_NTFS
Locked          : False

FullName        : C:\My Folder\virtualdisk1.vbox
Size            : 1034
Mechanism       : AES 256
Mounted         : False
MountLetter     : Y
AutomaticMount  : False
AccessMode      : Unspecified
Label           : Sample-Virtual-Disk-2
FileSystem      : eFSTYPE_NTFS
Locked          : False
```

Retrieve information about a Virtual Disk volume

```
C:\PS>$volume = New-SDSDisk 'C:\My Folder\virtualdisk.vbox' -Size 12
Mount-SDSDisk -Volume $volume -MountLetter Z
```

```
Get-SDSDisk -Drive Z
```

This command retrieves the specified Virtual Disk volume information. The Drive parameter is used.

```
FullName        : C:\Test\disk.vbox
Size            : 12
Mechanism       : AES 256
Mounted         : True
MountLetter     : Y
AutomaticMount  : True
AccessMode      : ReadWrite
Label           : disk
FileSystem      : eFSTYPE_NONE
Locked          : False
```

Retrieve information about all currently mounted Virtual Disk volumes

```
C:\PS>Mount-SDSDisk 'C:\My Folder\virtualdisk1.vbox'
Mount-SDSDisk 'C:\My Folder\virtualdisk2.vbox'
Get-SDSDisk
```

This command retrives information about all currently mounted Virtual Disk volumes.

```
FullName        : C:\My Folder\virtualdisk1.vbox
Size            : 12
Mechanism       : AES 256
Mounted         : True
MountLetter     : Z
AutomaticMount  : True
AccessMode      : ReadWrite
Label           : Sample-Virtual-Disk-1
FileSystem      : eFSTYPE_FAT12
Locked          : False
```

```
FullName        : C:\My Folder\virtualdisk2.vbox
Size            : 1034
Mechanism       : AES 256
Mounted         : True
MountLetter     : Y
AutomaticMount  : True
AccessMode      : ReadWrite
Label           : Sample-Virtual-Disk-2
FileSystem      : eFSTYPE_NONE
Locked          : False
```

B.11 Get-SDSFile

Summary

Retrieves information about one or more files encrypted with Stormshield Data File.

Description

The Get-SDSFile cmdlet retrieves information about one or more files encrypted with Stormshield Data File.

Parameters

Name	Required	Position	Default value	Pipeline	Wildcard characters	Description

-Path <String []>	true	1	true (ByPropertyName)	false	Specifies the path to one or more files encrypted with Stormshield Data File.

Inputs

System.String[]

You can pipe an array of one or more encrypted files.

Outputs

Stormshield.DataSecurity.Connector.File.SecureFile[]

This object represents an array of files transciphered by Stormshield Data File. The SecureFile represents a file encrypted with Stormshield Data File.

Notes

If the specified file path does not exist, an exception is raised.

Examples

Retrieve information about an encrypted file

```
C:\PS>Get-SDSFile 'C:\My Folder\Document.docx.sdsx'
```

This command retrieves information about the specified encrypted file.

```
Path                : C:\My Folder\Document.docx.sdsx
Encrypted           : True
OriginalFileName    : C:\My Folder\Document.docx
```

```
Size                : 154
Compressed          : False
Executable          : False
Mechanism           : AES 256
Author              : Alice Smith
Coworkers           : {recovery@mycompany, alicesmith@mycompany.com}
Certificates        : {Alice SMITH}
CertRetrievalStatus : SUCCEEDED
```

Retrieve information about several encrypted files

```
C:\PS>Get-SDSFile -Path 'C:\My Folder\Document.docx.sdsx','C:\My Folder\Document.xlsx.sdsx'
```

This command retrieves information about the specified encrypted files.

```
Path                : C:\My Folder\Document.docx.sdsx
Encrypted           : True
OriginalFileName    : C:\My Folder\Document.docx
Size                : 154
Compressed          : False
Executable          : False
Mechanism           : AES 256
Author              : Alice Smith
Coworkers           : {recovery@mycompany, alicesmith@mycompany.com}
Certificates        : {Alice SMITH}
CertRetrievalStatus : SUCCEEDED

Path                : C:\My Folder\Document.xlsx.sdsx
Encrypted           : True
OriginalFileName    : C:\My Folder\Document.docx
Size                : 312
Compressed          : False
Executable          : False
```

```
Mechanism           : AES 256
Author              : Alice Smith
Coworkers           : {alicesmith@mycompany.com}
Certificates        : {Alice SMITH}
CertRetrievalStatus : SUCCEEDED
```

B.12 Get-SDSTeamFile

Summary

Retrieves information about one or more files encrypted with Stormshield Data Team.

Description

The Get-SDSTeamFile cmdlet retrieves information about one or more files encrypted with Stormshield Data Team.

Parameters

Name	Required	Position	Default value	Pipeline	Wildcard characters	Description
-Path <String []>	true	1		true (ByPropertyName)	false	Specifies the path to one or more encrypted files. If one of the files does not exist, a FileNotFoundException if raised.

Inputs

System.String[]

You can pipe an array of strings containing one or more paths to encrytped files.

Outputs

Stormshield.DataSecurity.Connector.Team.FileInfoData[]

This object represents an array of files encrypted with Stormshield Data Team. The FileInfoData object represents a file encrypted with Stormshield Data Team. The FileInfoData object contains the following members: - Logical file size (excluding security header) - Physical file size (including security header) - Creator of the file - Size of security header - Flag that indicates if the file is encrypted or not. If the file is not encrypted, all members apart from physical size are irrelevant. - Encryption algorithm used - Owners and coworkers authorized for this file

Notes

This cmdlet does not require a user to be connected in order to be run.

Examples

Retrieve information about encrypted files

```
C:\PS>Get-SDSTeamFile -Path 'C:\My Secured Folder\Document.docx','C:\My Secured Folder\Document.xlsx'
```

This command retrieves information about the specified encrypted files.

```
FullName            : C:\My Secured Folder\Document.docx
LogicalSize         : 12596
```

```
PhysicalSize         : 16692
Creator              : Alice Smith
HeaderSize           : 4096
EncryptionMechanism : eEncrypterKeyType_AES256
Encrypted            : True
Coworkers            : {MyCompany Recovery, Alice Smith}

FullName             : C:\My Secured Folder\Document.docx
LogicalSize          : 8559
PhysicalSize         : 12655
Creator              : Alice Smith
HeaderSize           : 4096
EncryptionMechanism : eEncrypterKeyType_AES256
Encrypted            : True
Coworkers            : {MyCompany Recovery, Alice Smith}
```

Retrieve information about encrytped files

```
C:\PS>Get-ChildItem -Recurse | Where-Object { $_.PsIsContainer -eq $False } | Get-SDSTeamFile
```

This command recursively retrieves information about encrytped files in the current working directory.

```
FullName             : C:\My Secured Folder\Document.docx
LogicalSize          : 12596
PhysicalSize         : 16692
Creator              : Alice Smith
HeaderSize           : 4096
EncryptionMechanism : eEncrypterKeyType_AES256
Encrypted            : True
Coworkers            : {MyCompany Recovery, Alice Smith}

FullName             : C:\My Secured Folder\Document.docx
LogicalSize          : 8559
```

```
PhysicalSize         : 12655
Creator              : Alice Smith
HeaderSize           : 4096
EncryptionMechanism : eEncrypterKeyType_AES256
Encrypted            : True
Coworkers            : {MyCompany Recovery, Alice Smith}
```

Retrieve information about an encrypted file

```
C:\PS>Get-SDSTeamFile 'C:\My Secured Folder\Document.pdf'
```

This command retrieves information about the specified encrypted file.

```
FullName             : C:\My Secured Folder\Document.docx
LogicalSize          : 12477
PhysicalSize         : 16573
Creator              : Alice Smith
HeaderSize           : 4096
EncryptionMechanism : eEncrypterKeyType_AES256
Encrypted            : True
Coworkers            : {MyCompany Recovery, Alice Smith}
```

B.13 Get-SDSTeamRule

Summary

Retrieves information about one or more folders secured with Stormshield Data Team.

Description

The Get-SDSTeamRule cmdlet retrieves information about one or more folders encrypted with Stormshield Data Team.

Parameters

Name	Required	Position	Default value	Pipeline	Wildcard characters	Description
-Path <String []>	false	1		true (ByPropertyName)	false	Specifies the path to one or more secured folders. If one of the folders does not exist, a FileNotFoundException if raised. If this parameter is not specified, the current working folder is processed.

Inputs

System.String[]

You can pipe an array of strings containing one or more path to secured folders.

Outputs

Stormshield.DataSecurity.Connector.Team.RuleInfoData[]

This object represents an array of folders secured with Stormshield Data Team. The RuleInfoData object represents a folder secured with Stormshield Data Team. The RuleInfoData object contains the following information: - Path of the hidden SBoxTeam.sbt file. - Size of the hidden SBoxTeam.sbt file. - List of unencrypted files in the folder. - List of owners/coworkers allowed on this folder. - Flag that indicates if the folder is secured or not. If the folder is not secured, all members are irrelevant.

Notes

If no user is connected, an exception is raised. If a user is connected but locked, an exception is raised.

Examples

Retrieve information about several secured folders

```
C:\PS>Get-SDSTeamRule -Path 'C:\My Secured Folder 1','C:\My Secured Folder 2'
```

This command retrieves information about the specified secured folders.

```
FullName         : C:\My Secured Folder 1
SBTFile          : C:\My Secured Folder 1\SBoxTeam.sbt
SBTSize          : 4132
UnencryptedFiles : {}
Coworkers        : {MyCompany Recovery, Alice SMITH}
Secured          : True

FullName         : C:\My Secured Folder 2
SBTFile          : C:\My Secured Folder 2\SBoxTeam.sbt
SBTSize          : 4132
UnencryptedFiles : {}
Coworkers        : {MyCompany Recovery, Alice Smith}
Secured          : True
```

Retrieve information about a secured folder

```
C:\PS>Get-SDSTeamRule 'C:\My Secured Folder'
```

This command retrieves information about the specified secured folder.

```
FullName          : C:\My Secured Folder
SBTFile           : C:\My Secured Folder\SBoxTeam.sbt
SBTSize           : 4132
UnencryptedFiles  : {}
Coworkers         : {MyCompany Recovery, Alice SMITH}
Secured           : True
```

Retrieve information about each subfolders of current working folder

```
C:\PS>cd 'C:\My Secured Folder'
Get-ChildItem | Where-Object { $_.PSIsContainer -eq $True } | Get-SDSTeamRule
```

This command retrieves information about each subfolders of the current working folder.

```
FullName          : C:\My Secured Folder 1
SBTFile           : C:\My Secured Folder 1\SBoxTeam.sbt
SBTSize           : 4132
UnencryptedFiles  : {}
Coworkers         : {MyCompany Recovery, Alice SMITH}
Secured           : True

FullName          : C:\My Secured Folder 2
SBTFile           : C:\My Secured Folder 2\SBoxTeam.sbt
SBTSize           : 4132
UnencryptedFiles  : {}
Coworkers         : {MyCompany Recovery, Alice Smith}
Secured           : True
```

B.14 Get-SDSUser

Summary

Retrieves information about the currently connected user.

Description

The Get-SDSUser cmdlet retrieves information about the currently connected user.

Parameters

Name	Required	Position	Default value	Pipeline	Wildcard characters	Description

Inputs

Outputs

Stormshield.DataSecurity.Connector.Kernel.User

This object represents a Stormshield Data Security account.

Notes

If no user is connected, it returns nothing.

Examples

Retrieve currently connected user

```
C:\PS>Get-SDSUser
```

This command retrieves information about the currently connected user.

```
Id                    : alicesmith
Name                  : Alice Smith
Locked                : False
EmailAddresses        : {alice.smith@mycompany.com}
EncryptionCertificate : Alice Smith
SignatureCertificate  : Alice Smith
```

B.15 Import-SDSAddressBook

Summary

Imports a .p7b or .p7z file into the currently connected user's address book

Description

The Import-SDSAddressBook imports certificates from a .p7b or a .p7z file.

Parameters

Name	Required	Position	Default value	Pipeline	Wildcard characters	Description
-Path <String>	true	1		true (ByValue)	false	Specifies the path to the .p7b or .p7z file to import.

Inputs

System.String

You can pipe a string containing the path to .p7b or .p7z file to import.

Outputs

None

Notes

The .p7z file may contain personalized data related to certificates in addition to other information.

Examples

Import certificates

```
C:\PS>Import-SDSAddressBook C:\addressbook.p7b
```

This command imports certificates contained in the specified .p7b file into the address book of currently connected user. Trust chains, contacts and groups are imported if existing in .p7b file.

Restore address book content

```
C:\PS>Import-SDSAddressBook C:\addressbook.p7z
```

This command restores the address book of currently connected user to a previous state. Trusted chains, contacts, groups and personalized data are imported.

B.16 Lock-SDSUser

Summary

Locks a Stormshield Data Security session.

Description

The Lock-SDSUser cmdlet locks the current Stormshield Data Security session.

Parameters

Name	Required	Position	Default value	Pipeline	Wildcard characters	Description

Inputs

Outputs

void

Returns nothing.

Notes

If no user is connected, an exception is raised. If a user is already connected and locked, an exception is raised.

Examples

Locks session of the currently connected user

```
C:\PS>Lock-SDSUser
```

This command locks the session of the currently connected user.

B.17 Mount-SDSDisk

Summary

Mounts a Virtual Disk volume.

Description

The Mount-SDSDisk mounts a Virtual Disk volume.

Parameters

Name	Required	Position	Default value	Pipeline	Wildcard characters	Description
-Volume <Volume>	true	named		true (ByValue)	false	Specifies the Virtual Disk volume to be mounted.
-MountLetter <String>	false	named		false	false	Specifies drive unit mount letter to be used when mounting the Virtual Disk volume.
-AccessMode <AccessMode>	false	named	Unspecified	false	false	Specifies the access mode when mounting the Virtual Disk volume. Available access modes are: ReadWrite, ReadOnly and Unspecified. If access mode is Unspecified, the best available access is automatically selected.
-Path <String>	true	1		true (ByPropertyName)	false	Specifies the path to .vbox file to be mounted.

Inputs

System.String, Stormshield.DataSecurity.Connector.VirtualDisk.Volume, System.String, Stormshield.DataSecurity.Connector.VirtualDisk.AccessMode

You can pipe a string containing a path to .vbox file or a Virtual Disk volume.

Outputs

Stormshield.DataSecurity.Connector.VirtualDisk.Volume

This object represents a Virtual Disk volume.

Notes

If no user is connected, an exception is raised. If a user is connected but locked, an exception is raised.

Examples

Mount a Virtual Disk volume

```
C:\PS>Mount-SDSDisk 'C:\My Folder\virtualdisk.vbox'
```

This command mounts a Virtual Disk volume. The Path parameter is used.

```
FullName       : C:\My Folder\virtualdisk.vbox
Size           : 12
Mechanism      : AES 256
Mounted        : True
MountLetter    : Z
AutomaticMount : False
AccessMode     : ReadWrite
Label          : Sample-Virtual-Disk
FileSystem     : eFSTYPE_NONE
Locked         : False
```

Mount a Virtual Disk volume

```
C:\PS>$volume = New-SDSDisk 'C:\My Folder\virtualdisk.vbox' -Size 12
Mount-SDSDisk -Volume $volume
```

This command mounts a Virtual Disk volume. The Volume parameter is used.

```
FullName        : C:\My Folder\virtualdisk.vbox
Size            : 12
Mechanism       : AES 256
Mounted         : True
MountLetter     : Z
AutomaticMount  : False
AccessMode      : ReadWrite
Label           : Sample-Virtual-Disk
FileSystem      : eFSTYPE_NONE
Locked          : False
```

Mount a Virtual Disk volume

```
C:\PS>Mount-SDSDisk 'C:\My Folder\virtualdisk.vbox' -MountLetter Z -AccessMode ReadOnly
```

This command mounts a Virtual Disk volume. The Path parameter is used and the drive unit mount letter is explicitly specified. The Virtual Disk volume is mounted in read-only mode.

```
FullName        : C:\My Folder\virtualdisk.vbox
Size            : 12
Mechanism       : AES 256
Mounted         : True
MountLetter     : Z
AutomaticMount  : False
AccessMode      : ReadOnly
Label           : Sample-Virtual-Disk
FileSystem      : eFSTYPE_NONE
Locked          : False
```

B.18 New-SDSDisk

Summary

Creates one or more volumes encrypted with Stormshield Data Virtual Disk.

Description

The New-SDSDisk creates one or more Virtual Disk volumes.

Parameters

Name	Required	Position	Default value	Pipeline	Wildcard characters	Description
-Path <String[]>	false	1		true (ByValue)	false	Specifies the path to one or more .vbox files to create. Notes: - The file extension must be .vbox, otherwise a InvalidExtensionException exception is raised. - If not specified, the value specified in the Stormshield Data Security Suite configuration file will be used. - If the .vbox file already exists, a Stormshield.DataSecurity.Connector.VirtualDisk.FileAlreadyExistException exception is raised.

-Size <Int32>	false	named	false	false	Specifies the size of the Virtual Disk volume to create, in megabytes. The minimum value is 0, the maximum Value is 2097150. Notes: - If not specified or zero, the value specified in the Stormshield Data Security Suite configuration file will be used. If the configuration file does not contain this information, the size is fixed to 10% of the free space available on the hard drive. - If there is not enough space available on the hard drive, a Stormshield.DataSecurity.Connector.VirtualDisk.Exception exception is raised (E_SBD_NOT_ENOUGH_SPACE error code).

Inputs

System.String[], int

You can pipe an array of strings containing one or more path to .vbox files to create or the size of the Virtual Disk volumes to create.

Outputs

Stormshield.DataSecurity.Connector.VirtualDisk.Volume[]

This object represents an array of created Virtual Disk volumes. The Volume object represents a volume encrypted with Stormshield Data Virtual Disk.

Notes

The created Virtual Disk volumes are not formatted and can be formatted for any file system supported by the operating system. The Virtual Disk volumes need to be mounted prior to formatting it. If no user is connected, an exception is raised. If a user is connected but locked, an exception is raised.

Examples

Create two Virtual Disk volumes

```
C:\PS>New-SDSDisk -Path 'C:\My Folder\virtualdisk1.vbox','C:\My Folder\virtualdisk2.vbox'
```

This command creates two Virtual Disk volumes.

```
FullName        : C:\My Folder\virtualdisk1.vbox
Size            : 3086
Mechanism       : AES 256
Mounted         : False
MountLetter     : ?
AutomaticMount  : False
AccessMode      : Unspecified
Label           : Sample-Virtual-Disk-1
FileSystem      : eFSTYPE_NONE
Locked          : False

FullName        : C:\My Folder\virtualdisk2.vbox
Size            : 3076
Mechanism       : AES 256
Mounted         : False
MountLetter     : ?
AutomaticMount  : False
AccessMode      : Unspecified
Label           : Sample-Virtual-Disk-2
FileSystem      : eFSTYPE_NONE
Locked          : False
```

B.19 New-SDSTeamRule

Summary

Secures one or more folders with Stormshield Data Team.

Description

The New-SDSTeamRule cmdlet secures one or more folders with Stormshield Data Team.

Parameters

Name	Required	Position	Default value	Pipeline	Wildcard characters	Description
-Path <String[]>	false	1		true (ByPropertyName)	false	Specifies the path to one or more folders to secure.
-Owners <X509Certificate[]>	false	named		false	false	Specifies one or more owners certificates. The currently connected user is automatically added as owner and coworker. If one of the owners certificates is not also included in the coworkers certificates, an exception is raised.
-Coworkers <X509Certificate[]>	false	named		false	false	Specifies one or more coworkers certificates. The currently connected user is automatically added as owner and coworker.

-Force <SwitchParameter>	false	named	false	false	Specifies that the securing needs to be done by bypassing any confirmation request. The behavior is different according to the status of the certificate. If the status is warning, the rule is created with the certificate. If the status is error, the rule is created without the certificate. This parameter involves a answer "yes" to the following questions: Warning status: "Do you want to encrypt this file with this certificate anyway?" Error status: "If you continue, files will not be encrypted for this coworker."

Inputs

System.String[], Stormshield.DataSecurity.Connector.Common.X509Certificate[], Stormshield.DataSecurity.Connector.Common.X509Certificate[], SwitchParameter

You can pipe an array of strings containing one or more paths to folders that needs to be secured, the list of owners certificates, the list of coworkers certificates or a flag to force securing. You can pipe: An array of string containing one or more paths to the folders on which the Team Rule must be created, the list of owners's certificates, the list of coworkers's certificate or a flag to force creation

Outputs

Stormshield.DataSecurity.Connector.Team.RuleInfoData[]

This object represents an array of folders secured with Stormshield Data Team. The RuleInfoData object represents a folder secured with Stormshield Data Team. The RuleInfoData object contains the following information: - Path of the hidden SBoxTeam.sbt file - Size of the hidden SBoxTeam.sbt file. - List of unencrypted files in the folder. - List of owners/coworkers allowed on this folder. - Flag that indicates if the folder is secured or not. If the folder is not secured, all members are irrelevant.

Notes

If no user is connected, an exception is raised. If a user is connected but locked, an exception is raised. The currently connected user is automatically added to the list of owners. If the certificate's status is one of warning or error, a user confirmation is requested. The answer to this question is stored and is asked only once for each certificate during the Stormshield Data Security account session duration. The Force parameter answers by default to this confirmation request and the answer is also recorded.

Examples

Secures a folder

```
C:\PS>$owners = Get-SDSCertificate -EmailAddress robertmiller@mycompany.com

$coworkers = Get-SDSCertificate -EmailAddress robertmiller@mycompany.com,jodiefisher@mycompany.com

New-SDSTeamRule 'C:\My Secured Folder' -Owners $owners -Coworkers $coworkers
```

This command secures the specified folder. Robert Miller will be defined as owner. Jodie Fisher will be defined as coworker. Alice Smith is automatically added as owner because she is connected.

```
FullName          : C:\My Secured Folder
SBTFile           : C:\My Secured Folder\SBoxTeam.sbt
SBTSize           : 4132
UnencryptedFiles  : {}
Coworkers         : {MyCompany Recovery, Alice SMITH, Jodie FISHER, Robert Miller}
Secured           : True
```

B.20 Protect-SDSFile

Summary

Encrypts one or more files with Stormshield Data File.

Description

The Protect-SDSFile cmdlet encrypts one or more files with Stormshield Data File.

Parameters

Name	Required	Position	Default value	Pipeline	Wildcard characters	Description
-Path <String[]>	true	1		true (ByPropertyName)	false	Specifies the path to one or more files to encrypt.
-Coworkers <X509Certificate[]>	false	named		false	false	Specifies one or more coworkers certificates. The currently connected user is automatically added as coworker.

Inputs

System.String[], Stormshield.DataSecurity.Connector.Common.X509Certificate[]

You can pipe an array of strings containing one or more paths of the files that are to be encrypted or the list of coworkers certificates.

Outputs

Stormshield.DataSecurity.Connector.File.SecureFile[]

This object represents an array of files encrypted with Stormshield Data File. The SecureFile object represents a file encrytped with Stormshield Data File.

Notes

The currently connected user is automatically added to the list of coworkers. After the file is encrypted, the original file is securely deleted and the encrypted .sdsx file is created. If no user is connected, an exception is raised. If the status of one of the coworkers certificates is warning or error, no error will be reported and the file will be encrypted without these certificates.

Examples

Encrypt a file for the currently connected user

```
C:\PS>Protect-SDSFile 'C:\My Folder\Document.docx'
```

This command encrypts the specified files. The only coworker for this file will be the currently connected user.

```
Fullname             : C:\My Folder\Document.docx.sdsx
Encrypted            : True
OriginalFilename     : C:\My Folder\Document.docx
Size                 : 443940
Compressed           : False
Executable           : False
Mechanism            : AES 256
Author               : Alice Smith
Coworkers            : {recovery@mycompany, alicesmith@mycompany.com}
```

```
Certificates        : {Alice SMITH}
CertRetrievalStatus : SUCCEEDED
```

Encrypt a file for coworkers

```
C:\PS>$certificates = Get-SDSCertificate -Name 'Jodie FISHER'
Protect-SDSFile 'C:\My Folder\Document.docx' -CoWorkers $certificates
```

This command encrypts the specified files. Coworkers will be the currently connected user plus Jodie Fisher.

```
Fullname            : C:\My Folder\Document.docx.sdsx
Encrypted           : True
OriginalFilename    : C:\My Folder\Document.docx
Size                : 443940
Compressed          : False
Executable          : False
Mechanism           : AES 256
Author              : Alice Smith
Coworkers           : {recovery@mycompany, alicesmith@mycompany.com, jodiefisher@mycompany}
Certificates        : {Alice SMITH}
CertRetrievalStatus : SUCCEEDED
```

Encrypt all files in a folder

```
C:\PS>Get-ChildItem 'C:\My Folder\*.pdf' | Protect-SDSFile
```

This command encrypts all PDF files in the specified folder. The only coworker for this file will be the currently connected user.

```
Fullname             : C:\My Folder\Document1.pdf.sdsx
Encrypted            : True
OriginalFilename     : C:\My Folder\Document1.pdf
Size                 : 443940
Compressed           : False
Executable           : False
Mechanism            : AES 256
Author               : Alice Smith
Coworkers            : {recovery@mycompany.com, alicesmith@mycompany.com}
Certificates         : {Alice SMITH}
CertRetrievalStatus  : SUCCEEDED

Fullname             : C:\My Folder\Document2.pdf.sdsx
Encrypted            : True
OriginalFilename     : C:\My Folder\Document2.pdf
Size                 : 352561
Compressed           : False
Executable           : False
Mechanism            : AES 256
Author               : Alice Smith
Coworkers            : {recovery@mycompany.com, alicesmith@mycompany.com}
Certificates         : {Alice SMITH}
CertRetrievalStatus  : SUCCEEDED

Fullname             : C:\My Folder\Document3.pdf.sdsx
Encrypted            : True
OriginalFilename     : C:\My Folder\Document3.pdf
Size                 : 21538
Compressed           : False
Executable           : False
Mechanism            : AES 256
Author               : Alice Smith
Coworkers            : {recovery@mycompany, alicesmith@mycompany.com}
Certificates         : {Alice SMITH}
CertRetrievalStatus  : SUCCEEDED
```

B.21 Protect-SDSTeam

Summary

Encrypts all files contained in a folder secured by Stormshield Data Team.

Description

The Protect-SDSTeam cmdlet recursively encrypts all files contained in a folder secured by Stormshield Data Team. All the files will be encrypted with owners and coworkers defined at the folder level.

Parameters

Name	Required	Position	Default value	Pipeline	Wildcard characters	Description
-Path <String[]>	false	1		true (ByPropertyName)	false	Specifies the path to one or more folders to protect. If this parameter is not specified, the current working folder is protected. If the specified folder does not exist, a System.IO.DirectoryNotFoundException exception is raised.

Inputs

System.String[]

You can pipe an array of strings containing one or more paths to secured folders.

Outputs

Stormshield.DataSecurity.Connector.Team.OperationStatus[]

This object represents an array of statuses. The OperationStatus object represents the status of one encrypting operation.

Notes

If no user is connected, an exception is raised. If a user is connected but locked, an exception is raised.

Examples

Protect a folder recursively

```
C:\PS>Protect-SDSTeam 'C:\My Secured Folder'
```

This command protects the specified folder.

```
FileInfoData                                                          Status
------------                                                          ------
Stormshield.DataSecurity.Connector.Team.FileInfoData                eEIS_Encrypted
Stormshield.DataSecurity.Connector.Team.FileInfoData                eEIS_Encrypted
Stormshield.DataSecurity.Connector.Team.FileInfoData         eEIS_AlreadyEncrypted
Stormshield.DataSecurity.Connector.Team.FileInfoData                eEIS_Encrypted
```

B.22 Remove-SDSFileCoworker

Summary

Removes coworkers to one or more files encrypted with Stormshield Data File.

Description

The Remove-SDSFileCoworker cmdlet removes one or more coworkers to the coworker list of files encrypted with Stormshield Data File. It invokes transciphering mechanisms.

Parameters

Name	Required	Position	Default value	Pipeline	Wildcard characters	Description
-Path <String[]>	true	1		true (ByPropertyName)	false	Specifies the path to one or more files encrypted with Stormshield Data File.
-EmailAddress <String[]>	false	2		true (ByPropertyName)	false	Specifies one or more e-mail addresses to identify coworkers to remove from the encrypted file. Note: This parameters is not case sensitive.
-Coworkers <X509Certificate[]>	false	2		true (ByPropertyName)	false	Specifies one or more X.509 certificates to remove from the encrypted file.

Inputs

System.String[], System.String[], Stormshield.DataSecurity.Connector.Common.X509Certificate[]

You can pipe the list of files to be transciphered or the list of X.509 certificates to add.

Outputs

Stormshield.DataSecurity.Connector.File.SecureFile[]

This object represents an array of files transciphered by Stormshield Data File. The SecureFile represents a file encrypted with Stormshield Data File.

Notes

If no user is connected, an exception is raised. If a user is connected but locked, an exception is raised.

Examples

Remove coworkers from an encrypted file

```
C:\PS>Remove-SDSFileCoworker 'C:\My Folder\Document.docx.sdsx' -EmailAddress
jodiefisher@mycompany.com,robertmiller@mycompany.com
```

This command removes the specified coworkers from the encrypted file.

```
Path                : C:\My Folder\Document.docx.sdsx
Encrypted           : True
OriginalFileName    : C:\My Folder\Document.docx
Size                : 154
Compressed          : False
```

```
Executable          : False
Mechanism           : AES 256
Author              : Alice Smith
Coworkers           : {recovery@mycompany, alicesmith@mycompany.com}
Certificates        : {Alice}
CertRetrievalStatus : SUCCEEDED
```

B.23 Remove-SDSTeamRule

Summary

Removes security on a folder secured with Stormshield Data Team.

Description

The Remove-SDSTeamRule cmdlet removes security on a folder secured with Stormshield Data Team.

Parameters

Name	Required	Position	Default value	Pipeline	Wildcard characters	Description
-Path <String[]>	false	1		true (ByPropertyName)	false	Specifies the path to one or more folders to unsecure. If no folder is specified, the current working folder is unsecured. If the specified folder does not exist, a System.IO.DirectoryNotFoundException exception is raised.

Inputs

System.String[]

You can pipe an array of strings containing one or more paths to secured folders.

Outputs

void

Returns nothing.

Notes

If no user is connected, an exception is raised. If a user is connected but locked, an exception is raised. If the folder is not secured, an exception RuleNotFoundException is raised.

Examples

Remove security of a secured folder

```
C:\PS>Remove-SDSTeamRule 'C:\My Secured Folder'
```

This command unsecures the specified folder.

B.24 Set-SDSFileCoworker

Summary

Sets coworkers to one or more files encrypted with Stormshield Data File.

Description

The Set-SDSFileCoworker cmdlet sets one or more coworkers in the coworker list of files encrypted with Stormshield Data File. All the previous coworkers are replaced by the new ones. The currently connected user is automatically added to the coworkers list. It invokes transciphering mechanisms.

Parameters

Name	Required	Position	Default value	Pipeline	Wildcard characters	Description
-Path <String[]>	true	1		true (ByPropertyName)	false	Specifies the path to one or more files encrypted with Stormshield Data File.
-Coworkers <X509Certificate[]>	true	2		true (ByPropertyName)	false	Specifies one or more X.509 certificates to set in the encrypted file. Certificates will be added as coworkers.

Inputs

System.String[], Stormshield.DataSecurity.Connector.Common.X509Certificate[]

You can pipe the list of files to be transciphered or the list of X.509 certificates to set.

Outputs

Stormshield.DataSecurity.Connector.File.SecureFile[]

This object represents an array of files transciphered by Stormshield Data File. The SecureFile represents a file encrypted with Stormshield Data File.

Notes

If no user is connected, an exception is raised. If a user is connected but locked, an exception is raised.

Examples

Set coworkers for an encrypted file

```
C:\PS>$certificate = Get-SDSCertificate -EmailAddress jodiefisher@mycompany.com
Set-SDSFileCoworker 'C:\My Folder\Document.docx.sdsx' -Coworkers $certificate
```

This command sets the coworker Jodie Fisher for the file 'C:\Document.docx.sdsx'. The user Alice Smilth is automatically added because it is the currently connected user.

```
Path                : C:\My Folder\Document.docx.sdsx
Encrypted           : True
OriginalFileName    : C:\My Folder\Document.docx
Size                : 154
Compressed          : False
Executable          : False
Mechanism           : AES 256
Author              : Alice Smith
Coworkers           : {recovery@mycompany, alicesmith@mycompany.com, jodiefisher@mycompany.com}
Certificates        : {Alice SMITH, Jodie FISHER}
CertRetrievalStatus : SUCCEEDED
```

B.25 Unlock-SDSUser

Summary

Unlocks a Stormshield Data Security session.

Description

The Unlock-SDSUser cmdlet unlocks the current Stormshield Data Security session.

Parameters

Name	Required	Position	Default value	Pipeline	Wildcard characters	Description

-Password <String>	false	1	false	false	Specifies the password of the account. The password is the PIN of the smart card or USB token if applicable. Note: The password is case sensitive. If you enter your password incorrectly too many times (default is three tries), your account will be blocked. For example, with an account which three tries: First attempt, if the password is incorrect, a BadPasswordTwoAttemptsException exception is raised (two tries left). Second attempt, if the password is incorrect, a BadPasswordOneAttemptException exception is raised (one try left). Third attempt, if the password is incorrect, a BadPasswordAccountBlockedException exception is raised (account blocked). In interactive mode, this parameter is optional. If omitted or if the string is empty, the connection window opens up with an empty password field. If [Cancel] button is clicked in the connection window, an exception is raised (with E_LOGON_USER_CANCEL error code).
-SecurePassword <SecureString>	false	named	false	false	Specifies the password of the account. The password is the PIN of the smart card or USB token if applicable. Note: The password is case sensitive. This parameter allows the password to be specified in a secured manner. If you enter your password incorrectly too many times (default is three tries), your account will be blocked. For example, with an account which three tries: First attempt, if the password is incorrect, a BadPasswordTwoAttemptsException exception is raised (two tries left). Second attempt, if the password is incorrect, a BadPasswordOneAttemptException exception is raised (one try left). Third attempt, if the password is incorrect, a BadPasswordAccountBlockedException exception is raised (account blocked). In interactive mode, this parameter is optional. If omitted or if the string is empty, the connection window opens up with an empty password field. If [Cancel] button is clicked in the connection window, an exception is raised (with E_LOGON_USER_CANCEL error code).

Inputs

System.String, System.Security.SecureString

You can pipe the account password as a string or as a SecureString object.

Outputs

Stormshield.DataSecurity.Connector.Kernel.User

This object represents a Stormshield Data Security account.

Notes

If no user is connected, an exception is raised. If a user is already unlocked, an exception is raised.

Examples

Unlocks the currently connected user

```
C:\PS>Unlock-SDSUser password
```

This command unlocks the currently connected user.

```
Id                    : alicesmith
Name                  : Alice Smith
Locked                : False
EmailAddresses        : {alice.smith@mycompany.com}
EncryptionCertificate : Alice Smith
SignatureCertificate  : Alice Smith
```

Unlocks the currently connected user

```
C:\PS>Read-Host "password" -AsSecureString | ConvertFrom-SecureString | Out-File C:\secured-password.pwd
$secureString = (Get-Content C:\secured-password.pwd | ConvertTo-SecureString)
Unlock-SDSUser -SecurePassword $secureString
```

This command unlocks the currently connected user. A object of type SecureString is used for specifying the password in a secured manner.

```
Id                    : alicesmith
Name                  : Alice Smith
Locked                : False
EmailAddresses        : {alice.smith@mycompany.com}
EncryptionCertificate : Alice Smith
SignatureCertificate  : Alice Smith
```

B.26 Unprotect-SDSFile

Summary

Decrypts one or more files encrypted with Stormshield Data File.

Description

The Unprotect-SDSFile cmdlet decrypts one or more files encrypted with Stormshield Data File.

Parameters

Name	Required	Position	Default value	Pipeline	Wildcard characters	Description
-Path <String[]>	true	1		true (ByPropertyName)	false	Specifies the path to one or more files to decrypt.

Inputs

System.String[]

You can pipe an array of string containing one or more paths to files to decrypt.

Outputs

StormshieId.DataSecurity.Connector.File.SecureFile[]

This object represents an array of files decrypted by Stormshield Data File.

Notes

If no user is connected, an exception is raised. If a user is connected but locked, an exception is raised. If the currently connected user is not one of the file coworkers, an exception is raised.

Examples

Decrypt a list of files

```
C:\PS>Unprotect-SDSFile 'C:\My Folder\Document.docx.sdsx','C:\My Folder\Document.xlsx.sdsx'
```

This command decrypts the specified files.

```
Fullname             : C:\My Folder\Document.docx
Encrypted            : False
OriginalFilename     :
Size                 : 154
Compressed           : False
Executable           : False
Mechanism            :
Author               :
Coworkers            :
Certificates         :
CertRetrievalStatus  : ERROR_PLAIN_FILE

Fullname             : C:\My Folder\Document.xlsx
Encrypted            : False
OriginalFilename     :
Size                 : 1254
Compressed           : False
Executable           : False
Mechanism            :
Author               :
Coworkers            :
Certificates         :
CertRetrievalStatus  : ERROR_PLAIN_FILE
```

B.27 Unprotect-SDSTeam

Summary

Decrypts all files encrypted with Stormshield Data Team contained in a folder not secured with Stormshield Data Team.

Description

The Unprotect-SDSTeam cmdlet decrypts all files encrypted with Stormshield Data Team that lies in a folder not secured with Stormshield Data Team. When a folder is unsecured by using the Remove-SDSTeamRule cmdlet, its files are kept encrypted.

Parameters

Name	Required	Position	Default value	Pipeline	Wildcard characters	Description
-Path <String[]>	false	1		true (ByPropertyName)	false	Specifies the path to one or more folders to unprotect, or the path to one or more files to decrypt. If no folder is specified, the current folder is unprotected. If the specified path points to a nonexistant folder, a System.IO.DirectoryNotFoundException exception is raised. If the specified path points to a nonexistant file, a System.IO.FileNotFoundException exception is raised.
-Force <SwitchParameter>	false	named		false	false	Specifies that the decryption is forced, thus bypassing any confirmation request.

Inputs

System.String[], System.Management.Automation.SwitchParameter

You can pipe an array of strings containing one or more paths to folders or a flag to force decryption.

Outputs

StormshieId.DataSecurity.Connector.Team.OperationStatus[]

This object represents an array of statuses. The OperationStatus object represents the status of one encrypting operation.

Notes

If no user is connected, an exception is raised. If a user is connected but locked, an exception is raised. If the folder is secured, an exception is raised.

Examples

Decrypt several encrypted files

```
C:\PS>Unprotect-SDSTeam 'C:\My Unsecured Folder\Document.docx','C:\My Unsecured Folder\Document.xlsx'
```

This command decrypts the two specified files, as long as the parent folder is not unsecured.

```
FileInfoData                                                        Status
------------                                                        ------
Stormshield.DataSecurity.Connector.Team.FileInfoData        eEIS_Succeeded
Stormshield.DataSecurity.Connector.Team.FileInfoData        eEIS_Succeeded
```

Force files decryption

```
C:\PS>Unprotect-SDSTeam 'C:\My Unsecured Folder\Document.pdf -Force
```

This command decrypt the specified file, as long as the parent folder is not unsecured. No confirmation is prompted during the process.

```
FileInfoData                                                        Status
------------                                                        ------
Stormshield.DataSecurity.Connector.Team.FileInfoData        eEIS_Succeeded
```

documentation@stormshield.eu