GUIDE

STORMSHIELD DATA SECURITY ENTERPRISE

STORMSHIELD DATA CONNECTOR

Business applications encryption

Version 10.1

Document last update: March 29, 2022

Reference: sds-en-sd_connector-user_guide-v10

Table of contents

In the documentation, Stormshield Data Security Enterprise is referred to in its short form: SDS.

1. Getting started

Stormshield Data Connector is a component that comes with Stormshield Data Security. It allows driving the other components of the Suite.

Stormshield Data Authority Manager cannot be driven by Stormshield Data Connector.

Stormshield Data Connector comprises:

- The “Stormshield.DataSecurity.Connector.PowerShell” PowerShell module, which caters to administrators of IT environments who wish to run Stormshield Data Security through PowerShell scripts;
- .NET APIs, which cater to developers who wish to embed Stormshield Data Security features into their own products.

The use of the Stormshield Data Connector component requires thorough knowledge of PowerShell programming and/or a .NET language such as C#.

The PowerShell module and .NET APIs run in the same way, and commands used in both cases are identical. A .NET API is called up in the same way a PowerShell cmdlet command is written.

In this user guide, we refer to cmdlets in the context of a PowerShell script and APIs in the context of a .NET program.

2. Installing Stormshield Data Connector

This chapter provides information on Stormshield Data Connector requirements and installation.

2.1 Required configuration

For information on the required configuration of Microsoft Windows systems, refer to the section **Compatibility** of the 10.1 Stormshield Data Security Release Notes.

For Windows Server operating systems, a specific Server license is required for Stormshield Data Security.

2.2 Installing Stormshield Data Connector

Stormshield Data Security installation is global. The shipped product includes all the components of the software suite. Install the applications and components you need, according to the rights provided by the license key.

In order to run a specific component in the Suite with Stormshield Data Connector, this component has to be installed on the administrator's and user's workstations. For example, if you wish to use cmdlets or APIs from the Stormshield Data File component, install the File component on the workstations.

The installation procedure is described in the Stormshield Data Security *Installation and Implementation guide*.

2.3 Troubleshooting

2.3.1 Exception in the event of a missing SDS component

If the Stormshield Data Security component corresponding to a cmdlet or an API has not been installed on the workstation, an exception will be raised. For example, if the Stormshield Data File component has not been installed, the `Protect-SDSFile` cmdlet will raise the following exception:

```
Protect-SDSFile : This cmdlet depends on the component Stormshield Data
File which is not installed.
[...]
Protect-SDSFile c:\folder\document.docx
CategoryInfo : NotSpecified: (:) [], ComponentNotInstalledException
FullyQualifiedErrorId :
Stormshield.DataSecurity.Connector.PowerShell.ComponentNotInstalledExcepti
on
```

If a .NET API is used, the same exception will be raised.

2.3.2 Loading the Stormshield Data Connector component

PowerShell automatically loads the Stormshield Data Connector component the first time the available cmdlets are used.

If this is not the case, the module has to be imported using the command:

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

3. Using PowerShell cmdlets

3.1 Using and understanding cmdlets

Stormshield Data Connector comes with a ready-to-use PowerShell module containing cmdlets that allow running Stormshield Data Security. Each cmdlet enables the use of a function on the product.

For example, in order to retrieve an object corresponding to the logged in user, use the following cmdlet in a PowerShell console:

```
PS C:\> Get-SDSUser

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

The list of available cmdlets can be retrieved using the command

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

To get help and examples for each cmdlet, enter

```
Get-Help -Full <name-of-the-cmdlet>
```

More information about these cmdlets can be found in the List of the Stormshield Data Connector cmdlets.

TIP
On a 64-bit operating system, use a 64-bit PowerShell console.

3.2 Understanding the Windows PowerShell script execution policy

By default, Windows blocks the execution of PowerShell scripts.

The **Get-ExecutionPolicy** cmdlet allows knowing the current execution policy in force. The Windows PowerShell execution policies include the following :

- **Restricted**: No scripts can be run.
- **AllSigned**: Only scripts signed by a trusted publisher can be run.
- **RemoteSigned**: Scripts created locally will run but those downloaded from a network will not (unless they are digitally signed by a trusted publisher).
- **Unrestricted**: All Powershell scripts can be run.

The **Set-ExecutionPolicy** cmdlet allows changing the current user preference for the PowerShell execution policy. For example, if you want to execute a local PowerShell script, indicate the following cmdlet in a PowerShell window ("Run as administrator") :

```
Set-ExecutionPolicy RemoteSigned
```

4. Using .NET APIs

Stormshield Data Connector .NET APIs can be integrated into your applications in order to allow them to interact with Stormshield Data Security. In this section, the examples of code are given in C#.

4.1 Setting up a Visual Studio project

1. Create a C#.NET project, targeting .NET Framework 4.5.2. The .NET Framework 4.5.2 Developer Pack must have been installed beforehand.

   The .NET Framework 4.5.2 Developer Pack is only needed for compiling the application using Stormshield Data Connector. Running an application that uses Stormshield Data Connector does not require it.
2. Add the following reference to your project: *Stormshield.DataSecurity.Connector*.

This assembly is found in the Stormshield Data Security installation folder: by default it is **C:\Program Files\Arkoon\Security BOX\Connector**.

After that, the *Stormshield.DataSecurity.Connector* namespace can be used to benefit from its functionalities.

Some other namespaces (found in the *Stormshield.DataSecurity.Connector* assembly) can also be useful to handle some Stormshield Data Security particular objects. These namespaces can be found in the given examples throughout this document.

In some cases, an interoperability DLL will need to be added as a reference to your project. These assemblies can be found in the Stormshield Data Security installation folder.

On a 64-bit operating system, the .NET executable file has to be compiled for the 64-bit platform:

1. Go to the properties of the Visual Studio project, under the section "Build".
2. Select "Platform target = x64" or unselect "Prefer 32-bit" if you wish to remain in "Platform target Any CPU".

4.2 Using .NET APIs

Stormshield Data Connector .NET APIs are used in the same way as all cmdlets provided by the PowerShell module. The *Stormshield.DataSecurity.Connector* namespace contains all the objects useful to interact with Stormshield Data Security.

The *Stormshield.DataSecurity.Connector.API* class is the entry point to use the APIs. This is an object that implements the IDisposable interface. Once created, you can use the *Execute* method to call an API.

A .NET API is called up in the same way a PowerShell cmdlet command is called up. Two overloads of the *Execute* method are available.

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

This overload is used to simply call a PowerShell cmdlet that involves strings only (no .NET objects). The entire cmdlet (with arguments) must be passed in the string parameter.

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

This overload is used to call a PowerShell cmdlet that needs one or more .NET objects to be passed as arguments. The name of the cmdlet must be passed as the first parameter. The cmdlet arguments must be passed as a KeyValuePair array (refer to the use case below).

The returned object[] functions as follows:

- It is equal to "null" if the API does not return anything;
- It is equal to a table that may contain one or several objects if the API returns one or several results;
- The elements of the table can never be "null".

4.3 Examples

4.3.1 First overload of the *Execute* method

In this example, we attempt to retrieve the currently connected user.

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

If the returned array is not empty and the first item is not "null", the user has been successfully retrieved. The first item can then be cast to a User class, from the *Stormshield.DataSecurity.Connector.Kernel* namespace.

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

If the PowerShell cmdlet returns several objects, they are all available in the objects array.

When needed, it is the responsibility of the caller to make sure that the objects array is not empty and items are not "null".

4.3.2 Second overload of the *Execute* method

This example shows how the KeyValuePair table is used to encrypt several files:

```
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);
```

Each parameter has to be added to the KeyValuePair table of objects, with the key being the name of the cmdlet's parameter.

4.4 Running shipped code samples

Some sample projects are provided with a Visual Studio 2013 Solution.

Prerequisites:

- 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)

The .NET Framework 4.5.2 Developer Pack is only needed for compiling the application using Stormshield Data Connector. Running an application that uses Stormshield Data Connector does not require it.

Each project included in this solution demonstrates the use of a particular API.

4.5 .NET API usage scenarios

NOTE
In the following examples, the error scenarios are not managed.

4.5.1 Handling user connection

Connecting a user

The Connect-SDUser API allows connecting a user to 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);
}
```

Checking the connection status

Get-SDSUser API makes it possible to find out the status of the current Stormshield Data Security session (connected or locked):

```
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");
}
```

Locking a session

The Lock-SDSUser API allows locking a current Stormshield Data Security session:

```
using Stormshield.DataSecurity.Connector;
using (API api = new API())
{
  // first ensure that a user is connected
  api.Execute("Lock-SDSUser");
}
```

Unlocking a session

The Unlock-SDSUser API allows unlocking a current Stormshield Data Security session:

```
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);
}
```

Logging off a user

The Disconnect-SDUser API allows disconnecting the current user:

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

4.5.2 Interacting with the trusted address book

A Stormshield Data Security user must be connected in order to perform the operations described below (see Handling user connection). An exception is raised if no user is connected.

Retrieving certificates

The Get-SDSCertificate API allows retrieving one or several certificates found in the directory.

```
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
The retrieval of certificates by e-mail addresses relies only on the **E-mail address** field available in the certificate's *General* tab. E-mail addresses contained in the details of the certificate will not be used.

It is possible to retrieve several certificates at once, by using the PowerShell list syntax:

```
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;
}
```

If the UpdateStatus option was not selected during the command, the certificates will be retrieved without the status data. In order to get updates for this field, you will need to make an explicit request. The command will then take a longer time to send back certificates.

Exporting the contents of the address book

The contents of the trusted address book can be backed up in a file so that it can be restored later.

The Export-SDSAddressBook API allows exporting part of the contents of a user's address book.

The following example shows how to export in a P7B file certificates, their parent-child relationship as well as contacts and groups:

```
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;
}
```

The whole address book including the customization of certificates can be exported using the Backup-SDSAddressBook API.

The following example allows backing up the whole address book in a P7Z file:

```
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;
}
```

The *.p7z* file allows backing up customized certificate data. However, this format is not compatible with versions of Stormshield Data Security older than version 9.1.

Importing the contents of the address book

The contents of the address book can be imported using the following code. The example shown here uses a P7Z file, but any format that is compatible with the Suite's address book will work.

```
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 Interacting with Stormshield Data File

Getting information about files

The Get-SDSFile API can be used to retrieve some information about a file, encrypted or not.

```
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;
}
```

The SecureFile object contains the following information about the file:

- Whether the file is encrypted
- The list of recipients (email addresses, if file is encrypted). Information read from the header.

Multiple file paths can be passed to the Get-SDSFile API, as a PowerShell list:

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

In this case, the returned objects array contains two items, representing the information for each file passed.

No user needs to be connected to retrieve basic file information. To retrieve more details (regarding certificates), a user must be connected.

Encrypting a file

The Protect-SDSFile API can be used to encrypt a file with Stormshield Data File. A list of certificates must be retrieved from the user's address book (see Retrieving certificates).

First of all, a Stormshield Data Security user must be connected to encrypt any file (see Handling user connection). An exception is raised if no user is connected.

The following example shows how to encrypt a file for several coworkers by providing their certificates:

```
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;
}
```

To protect multiple files at once:

```
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)
};
```

Decrypting a file

The Unprotect-SDSFile API can be used to decrypt a file with Stormshield Data File.

A Stormshield Data Security user must be connected to decrypt any file (see Handling user connection). An exception is raised if no user is connected.

```
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;
}
```

Adding coworkers for secure files

The Add-SDSFileCoworker API allows adding one or several coworkers to one or several files encrypted with Stormshield Data File.

A Stormshield Data Security user must be connected in order to add coworkers (see Handling user connection). An exception is raised if no user is connected.

```
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;
}
```

Modifying the list of coworkers associated with secure files

The Set-SDSFileCoworker API allows replacing all coworkers associated with one or several files encrypted with Stormshield Data File by one or several other coworkers.

A Stormshield Data Security user must be connected in order to add coworkers (see Handling user connection). An exception is raised if no user is connected.

```
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;
}
```

Deleting coworkers associated with secure files

The Remove-SDSFileCoworker API allows deleting a set of coworkers associated with one or several files encrypted with Stormshield Data File.

A Stormshield Data Security user must be connected in order to add coworkers (see Handling user connection). An exception is raised if no user is connected.

```
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;
}
```

Coworkers that are identified either by their e-mail addresses or their certificates can be deleted. If they are being deleted based on their e-mail addresses, an exception will be raised if the address of a coworker is associated with a missing certificate.

4.5.4 Interacting with Stormshield Virtual Disk

A Stormshield Data Security user must be connected in order to perform the operations described below (see Handling user connection). An exception is raised if no user is connected.

Creating a volume

The New-SDSDisk API allows creating an encrypted virtual disk volume.

The following example shows how to create a 42 MB volume:

```
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;
}
```

In this example, the volume created is neither formatted nor mounted.

Getting information about a volume

The Get-SDSDisk API allows retrieving information about a virtual disk volume.

```
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;
}
```

Mounting a volume

The Mount-SDSDisk API allows mounting a virtual disk volume.

```
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;
}
```

Dismounting a volume

The Dismount-SDSDisk API allows dismounting a virtual disk volume.

```
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;
}
```

Enabling the automatic mounting of a volume

The Enable-SDSDiskAutomaticMount API allows enabling the automatic mounting of a virtual disk volume.

```
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;
}
```

Disabling the automatic mounting of a volume

The Disable-SDSDiskAutomaticMount API allows disabling the automatic mounting of a virtual disk volume.

```
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 Interacting with Stormshield Data Team

Creating a rule on a folder

A Stormshield Data Security user must be connected in order to create a rule (see Handling user connection). An exception is raised if no user is connected.

The New-SDSTeamRule cmdlet allows creating a rule on one or several folders with Stormshield Data Team.

The following example shows how to create a rule for several coworkers:

```
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;
}
```

The connected user who creates the rule will automatically be entered as the owner (Owners parameter). If no users are connected, then the creation of the rule will fail.

Reading the rule associated with a folder

A Stormshield Data Security user must be connected in order to read a rule (see Handling user connection). An exception is raised if no user is connected.

The Get-SDSTeamRule cmdlet allows reading the rule associated with one or several folders with 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;
}
```

Reading information from a file

The Get-SDSTeamFile cmdlet allows reading the Team information of one or several files with 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;
}
```

Applying a rule on a folder

A Stormshield Data Security user must be connected in order to apply a rule (see Handling user connection). An exception is raised if no user is connected.

The Protect-SDSTeam cmdlet allows applying a rule to one or several folders with Stormshield Data Team.

This rule must be created beforehand.

```
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;
  }
}
```

Deleting a rule associated with a folder

A Stormshield Data Security user must be connected in order to delete a rule (see Handling user connection). An exception is raised if no user is connected.

The Remove-SDSTeamRule cmdlet allows deleting a rule associated with one or several folders with Stormshield Data Team.

This rule must be created beforehand.

```
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));
}
```

Unsecuring a folder or file protected with a rule.

A Stormshield Data Security user must be connected to unsecure a folder or file (see Handling user connection). An exception is raised if no user is connected.

The Unprotect-SDSTeam cmdlet allows unsecuring one or several files or folders with Stormshield Data Team.

This rule must be deleted beforehand.

```
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 Using .NET APIs from a C++ program

To run Stormshield Data Security using a .NET API from a program written in C or C++, you will need to set up a bridge between both technologies by writing code in managed C++ ("C++/CLI wrapper") that will allow calling the .NET code from C or C++ code.

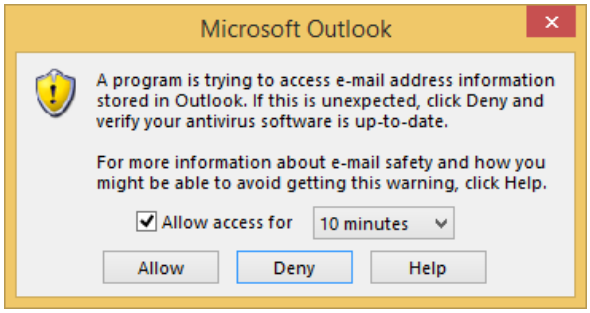
5. Running Stormshield Data Mail Outlook Edition

Stormshield Data Mail Outlook Edition can be run using a PowerShell script or a .NET program to send and read encrypted and/or signed messages.

5.1 Configuring Microsoft Outlook

By default, Microsoft Outlook restricts the execution of external programs.

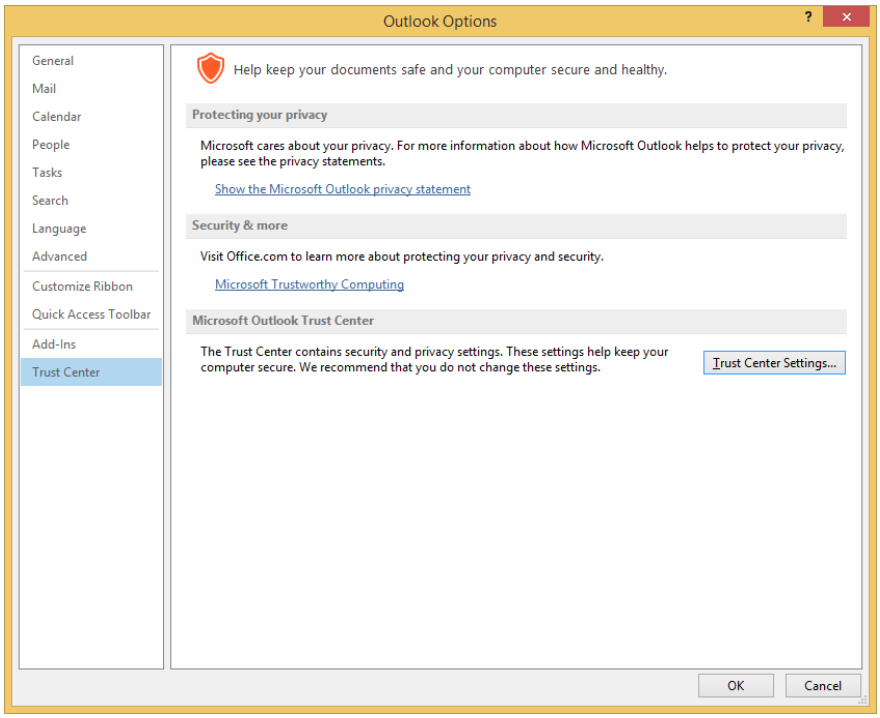
When you execute a PowerShell script or .NET program using Microsoft Outlook properties, the following pop up window appears:



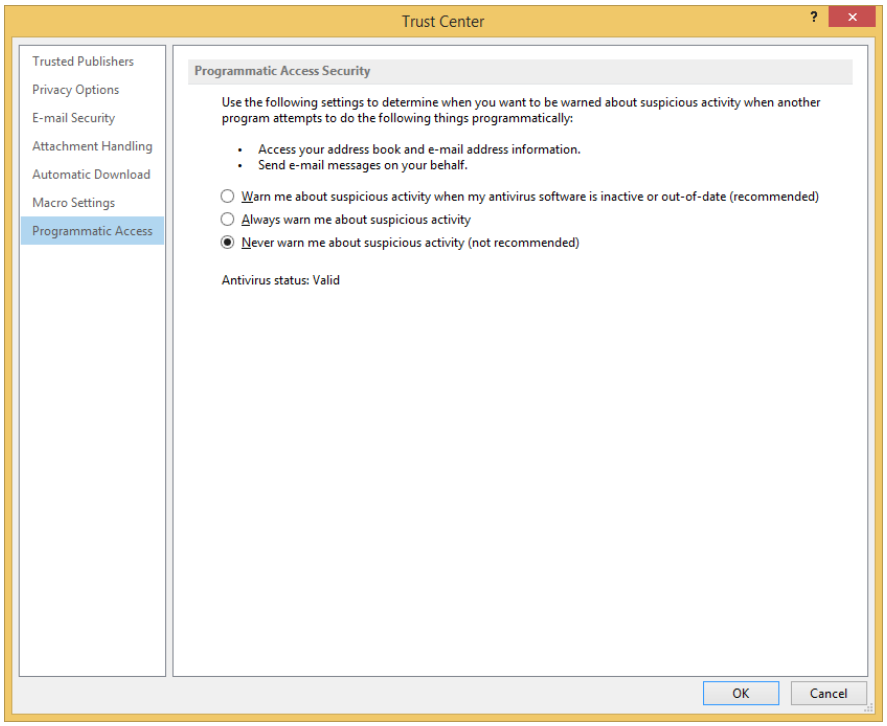
Click **Allow** to enable the PowerShell script or .NET program to run Microsoft Outlook. Access is allowed for a specified duration (1, 2, 5 or 10 minutes).

To execute a PowerShell script or .NET program using Microsoft Outlook and bypass this pop up window, configure Microsoft Outlook to accept being run by other programs:

1. Run Microsoft Outlook as administrator.
2. Open the Trust Center in the Outlook options and click **Trust Center Settings**:



3. Select **Never warn me about suspicious activity** in the *Programmatic Access* tab.



It is possible to configure a registry key on each workstation to automatically allow the execution of external programs in Microsoft Outlook. For Microsoft Outlook 2019 and Office 365, the registry key is:

- Microsoft Outlook 32 bits or 64 bits:
  HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Office\16.0\Outlook\Security
  DWORD: ObjectModelGuard
  Value: 0, 1 or 2
- Microsoft Outlook 32 bits running on 64-bit systems:
  HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\Office\16.0\Outlook\Security
  DWORD: ObjectModelGuard
  Value: 0, 1 or 2

The ObjectModelGuard values depend on the **Programmatic Access Security** parameters:

- **Warn me about suspicious activity when my antivirus software is inactive or out-of-date (recommended)** → Value **0**
- **Always warn me about suspicious activity** → Value **1**
- **Never warn me about suspicious activity (not recommended)** → Value **2**

5.2 Configuring Stormshield Data Mail Outlook Edition

In the normal operating mode of the Stormshield Data Mail Outlook Edition add-in, the EnableScripting value of the registry key **HKLM\SOFTWARE\Arkoon\Security BOX Enterprise\Properties\Mail** is disabled (value = 0). Therefore, when encrypted and/or signed messages are sent or read, if errors occur (issues with the certificate for example), dialog boxes that require the user's manual response will appear.

To run the add-in in "script" mode, i.e., from a PowerShell script or a .NET program, this registry value needs to be enabled (value = 1). As such, when errors occur, dialog boxes requiring the user's response will no longer appear.

If this value has not been enabled and the add-in is run in "script" mode, the dialog boxes that may open would prevent the encrypted and/or signed message from being sent or read. Potential errors that may occur during the process will be logged in the file *%TEMP%\Arkoon.SecurityBox.Mail.Scripting.log*.

After the process of sending or reading encrypted and/or signed messages has finished running, the value needs to be disabled where necessary in order to switch back to a normal operating mode of the Stormshield Data Mail Outlook Edition add-in.

The default value of the EnableScripting registry value is 0.

5.3 Interacting with Stormshield Data Mail Outlook Edition

The Stormshield Data Mail Outlook Edition module processes outgoing messages in order to encrypt and/or sign them and incoming messages to decrypt and/or read their signatures according to the properties assigned to the message.

5.3.1 Sending an encrypted and/or signed message

Create a PowerShell script or .NET program that will use the *Outlook Object Model* programming interface.

Add the following properties to the message created:

- **SDSEncrypt** to encrypt an e-mail.
- **SDSSign** to sign an e-mail.

The message will be encrypted with the certificates found in the address book of the current Stormshield Data Security user. If no user is connected then a connection pop-up window displays.

The message will be signed by the user logged on to Stormshield Data Security, using his private signature key.

For more information about creating a new user property in the UserProperties collection and about the security behavior of the Outlook Object Model, refer to the Microsoft Developer Network website.

The following examples show the two methods for opening a Microsoft Outlook instance and sending a signed and encrypted message.

The PowerShell script and .NET program automatically send an encrypted and signed message to the recipient john.doe@mycompany.com. This message will also be encrypted for the sender.

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
```

The script above does not take into account the release of COM objects that have been instantiated from Microsoft Outlook. They need to be released using:

```
[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
      }
    }
}
```

The program above does not take into account the release of COM objects that have been instantiated from Microsoft Outlook. They need to be released using:

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

5.3.2 Reading an encrypted and/or signed message

When a message is read in Microsoft Outlook through a PowerShell script or a .NET program, the Stormshield Data Outlook Edition add-in will decrypt the message seamlessly. The MailItem object retrieved in this manner will be in plain text and both properties will allow determining whether the message had been encrypted and/or signed:

- **SDSReadEncrypted** if the message was encrypted
- **SDSReadSigned** if the message was signed

The following examples show the two methods for opening a Microsoft Outlook instance and reading a message from the inbox.

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
```

The script above does not take into account the release of COM objects that have been instantiated from Microsoft Outlook. They need to be released using:

```
[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
    }
  }
}
```

The program above does not take into account the release of COM objects that have been instantiated from Microsoft Outlook. They need to be released using:

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

6. Disabling Stormshield Data Connector

If the Stormshield Data Connector component has been installed but you do not wish to use it, you may disable it.

To disable Stormshield Data Connector, you can delete the **Read & Execute** permission from the following file:

```
<installation folder>\Connector\Modules
↳ Stormshield.DataSecurity.Connector.PowerShell\
↳ Stormshield.DataSecurity.Connector.PowerShell.dll
```

7. Restriction

The Stormshield Data Connector component is not "Thread-Safe". The cmdlet/API user must therefore take precautions to protect his code using mutual exclusion structures ("Mutex") wherere necessary.

Annexe A. Example of a scenario

The same scenario is set out here in PowerShell script and .NET programming.

The administrator of an environment of machines wants a specific folder to be encrypted at all times on user workstations.

When a user logs on to his Windows session, the administrator wishes to:

- Connect the user to his Stormshield Data Security account (in interactive mode);
- Ensure that a given folder has been secured with the Security Data Team module (and create it if necessary);
- Inform the user of what has taken place.

The folder must always be secured using a Team rule and encrypted files.

This scenario can be implemented in the same way in both modes of using the Stormshield Data Connector module.

In both implementations, the configured folder is named "Secured" on the user's desktop.

The administrator can run the PowerShell script or the .NET program when the user's Windows session starts.

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());
      }
    }
  }
}
```

Appendix B. List of the Stormshield Data Connector cmdlets

This appendix provides a description and information about each Stormshield Data Connector cmdlet.

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