

# ADSDotNET Programmierleitfaden

ADSDotNET Version: 0.1.0  
Version des Dokuments: 1.0

## Einleitung

ADSDotNET ist eine Programmbibliothek, die es ermöglicht in .NET-basierten Programmiersprachen wie C# oder Visual Basic .NET auf Alternate Data Streams (ADS) zuzugreifen. Ohne diese DLL wäre es notwendig auf P-Invoke zurückzugreifen oder C++/CLI zu verwenden um mit ADS arbeiten zu können und gleichzeitig nicht auf die Vorzüge einer .net-basierten Sprache verzichten zu müssen. ADSDotNET ist übrigens in C++/CLI entwickelt worden.

## Systemvoraussetzungen

- Windows mit installiertem .NET Framework 2.0

## ADSDotNET verwenden

In Visual Studio muss die DLL zunächst als Reference (am besten über den Solution Explorer) eingebunden werden. Danach scheint sie unter dem Namen ADSDotNET im Ordner References auf und kann sofort verwendet werden!

Im Namespace `AdsDotNet` ist die Klasse `AdsHandler` zu finden. Um die Alternate Data Streams einer Datei oder eines Ordners bearbeiten zu können, muss zunächst einmal ein neues Objekt des Typs `AdsHandler` erzeugt werden, wobei dem Konstruktor der Name der Datei /des Ordners übergeben werden muss, dessen ADS abgefragt bzw. verändert werden sollen. Diese Datei bzw. dieser Ordner wird im Folgenden auch als Wurzelement bezeichnet. An das Wurzelement können alternative Datenströme „angeheftet“ werden. An dem erhaltenen Objekt können dann die Operationen wie `AddAds` oder `DeleteAds` angewendet werden. Um die verwendeten Ressourcen wieder freizugeben, ist es notwendig einen mit `OpenAds` geöffneten Stream mit `CloseAds` wieder zu schließen.

Es folgt nun eine Beschreibung der öffentlichen Methoden der Klasse `AdsHandler` inklusive Beispiele.

## Konstruktor

Erstellt ein neues Objekt des Typs `AdsHandler`, welches eine Datei oder einen Ordner repräsentiert, auf die in Folge ADS-Operationen angewendet werden können.

C++/CLI-Deklaration: `AdsHandler(String ^filename);`  
Parameter: `filename:` Der Name der Datei oder des Ordners, dessen ADS abgerufen oder verändert werden sollen.

Beispiel 1: `AdsManager adsman = new AdsManager("C:\\texte\\textdatei.txt");`  
Beispiel 2: `AdsManager adsman = new AdsManager("C:\\texte");`

Anmerkung: Wird ein leerer String übergeben - `new AdsManager("");` -, wird eine `CommandNotAllowedException` geworfen. Wenn die angegebene Datei bzw. der angegebene Ordner nicht existiert, wird eine `FileNotFoundException` geworfen.

## AddAds

Fügt der Datei / dem Ordner einen neuen leeren ADS hinzu.

C++/CLI-Deklaration: `void AddAds(String ^adsname);`  
Parameter: `adsname:` Der Name des zu erstellenden ADS.

Beispiel: `adsman.AddAds("mein_stream");`

Anmerkung: Wird ein leerer String übergeben - `adsman.AddAds("")` -, wird eine `CommandNotAllowedException` geworfen. Existiert der angegebene ADS bereits, geschieht nichts.

## DeleteAds

Löscht einen bestimmten ADS.

C++/CLI-Deklaration: `void DeleteAds(String ^adsname);`  
 Parameter: adsname: Der Name des zu löschenden ADS.

Beispiel: `adsman.AddAds("mein_stream");`

Anmerkung: Wird ein leerer String übergeben - `adsman.AddAds("")` -, wird, falls es sich beim Wurzelement um eine Datei handelt, wird die Datei inklusive all seiner ADS gelöscht. Handelt es sich beim Wurzelement um einen Ordner, wird eine `WinAPIException` geworfen.

## AdsExists

Liefert `true` zurück, wenn ein ADS mit dem angegebenen Namen existiert, ansonsten `false`.

C++/CLI-Deklaration: `bool AdsExists(String ^adsname);`  
 Parameter: adsname: Auf die Existenz eines Streams mit diesem Namen wird geprüft.

Beispiel: `if(adsman.AdsExists("stream334")) { ... }`

Anmerkung: Wird ein leerer String übergeben - `adsman.AdsExists("")` -, bezeichnet dies das Wurzelement.

## GetAdsList

Ermittelt die Namen aller an eine Datei / einen Ordner angehängten ADS. Es wird eine Liste von Strings mit den Namen der gefundenen ADS zurückgeliefert.

C++/CLI-Deklaration: `System::Collections::Generic::List<System::String^> ^GetAdsList();`

Beispiel: `List<String> adslist = adsman.GetAdsList();`

## OpenAds

Öffnet einen ADS zum Bearbeiten oder Lesen seines Inhaltes. Es wird ein `FileStream` zur Verfügung gestellt mit dem genau so gearbeitet werden kann wie mit normalen Dateien.

C++/CLI-Deklaration: `System::IO::FileStream ^OpenAds(String ^adsname);`  
 Parameter: adsname: Der Name des zu öffnenden ADS.

Beispiel: 

```
AdsHandler adsman_folder = new AdsHandler("testfolder");
adsman_folder.AddAds("asd");
System.IO.FileStream fs = adsman_folder.OpenAds("asd");
System.IO.StreamWriter sw = new System.IO.StreamWriter(fs);
sw.WriteLine("Hello ADS!");
sw.Close();
adsman_folder.CloseAds("asd");
```

Anmerkung: Existiert der zu öffnende Stream nicht, wird eine `StreamNotFoundException` geworfen. Wenn ein leerer String übergeben wird, wird ein `FileStream` auf das Wurzelement zurückgeliefert. Handelt es sich beim Wurzelement nicht um eine Datei, wird eine `WinAPIException` geworfen. Der `FileStream` wird innerhalb der Methode `CloseAds` geschlossen und muss daher nicht mit `fs.Close()` vom Benutzer geschlossen werden.

## AdsIsOpened

Liefert `true` zurück, wenn ein ADS mit dem angegebenen Namen momentan geöffnet ist, ansonsten `false`. Der Stream muss mittels der Methode `OpenAds` geöffnet worden sein. Daher liefert `AdsIsOpened` ausschließlich `true` zwischen den Methoden `OpenAds` und `CloseAds` (unabhängig davon, ob der `FileStream` eventuell schon vor `CloseAds` manuell mit `fs.Close()` geschlossen wurde).

C++/CLI-Deklaration: `bool AdsIsOpened(String ^adsname);`  
Parameter: `adsname:` Der Name des Streams, der darauf geprüft werden soll, ob er geöffnet ist.  
Beispiel: `if (adsman_folder.AdsIsOpened("stream")) { ... }`

## CloseAds

Schließt einen zuvor mit `OpenAds` geöffneten ADS. Der von `OpenAds` erhaltene `FileStream` wird mit dem Aufruf dieser Methode geschlossen.

C++/CLI-Deklaration: `void CloseAds(String ^adsname);`  
Parameter: `adsname:` Der Name eines geöffneten ADS, der geschlossen werden soll.  
Beispiel: (siehe `OpenAds`)

## GetCompleteFilename

Baut aus dem Dateinamen und dem Streamnamen einen Namen der Form `X:\pfad\datei.txt:streamname` bzw. `X:\pfad:streamname`.

C++/CLI-Deklaration: `System::String ^GetCompleteFilename(String ^adsname);`  
Parameter: `adsname:` Ein beliebiger Name, der als Streamname verwendet werden soll.  
Beispiel: `string filename = adsman.GetCompleteFilename("stream");`  
Anmerkung: Es muss kein Stream des angegebenen Namens existieren.