Erstellen sicherer ASP.NET-Anwendungen

Authentifizierung, Autorisierung und sichere Kommunikation

Auf der Orientierungsseite finden Sie einen Ausgangspunkt und eine vollständige Übersicht zum *Erstellen sicherer ASP.NET-Anwendungen*.

Zusammenfassung

Nachstehend wird erläutert, wie die DPAPI von einer ASP.NET-Webanwendung oder einem ASP.NET-Webdienst zum Verschlüsseln von vertraulichen Daten verwendet wird. In der nachstehend beschriebenen Vorgehensweise wird die DPAPI in Verbindung mit dem Benutzerspeicher eingesetzt, was die Verwendung einer "Out-of-Process"-Komponente von Enterprise Services voraussetzt.

Vorgehensweise: Verwenden von DPAPI (Benutzerspeicher) von ASP.NET aus mit Enterprise Services

In Webanwendungen müssen häufig sicherheitsrelevante Daten wie Datenbank-Verbindungszeichenfolgen und Anmeldeinformationen von Dienstkonten in Konfigurationsdateien gespeichert werden. Aus Sicherheitsgründen sollten solche Informationen niemals im Klartext gespeichert und vor dem Speichern immer verschlüsselt werden.

Diese Vorgehensweise beschreibt, wie die Data Protection API (DPAPI) von einer ASP.NET-Anwendung aus mit Enterprise Services verwendet wird.

Hinweise

- Die DPAPI kann entweder mit dem Computerspeicher oder mit dem Benutzerspeicher arbeiten (was ein geladenes Benutzerprofil voraussetzt). Standardmäßig verwendet die DPAPI den Benutzerspeicher, Sie können jedoch auch festlegen, dass der Computerspeicher verwendet werden soll, indem Sie das Flag CRYPTOPROTECT_LOCAL_MACHINE an die DPAPI-Funktionen übergeben.
- Mit der Verwendung des Benutzerprofils (wie in dieser Vorgehensweise gezeigt) wird eine zusätzliche Sicherheitsschicht eingezogen, denn hiermit wird der Zugriff auf die geheimen Daten weiter eingeschränkt. Nur der Benutzer, der die Daten verschlüsselt, kann sie auch wieder entschlüsseln. Allerdings erfordert die Verwendung des Benutzerprofils zusätzlichen Entwicklungsaufwand, wenn die DPAPI von einer ASP.NET-Anwendung verwendet werden soll, denn Sie müssen explizite Schritte zum Laden und Entladen des Benutzerprofils unternehmen (da ASP.NET Benutzerprofile nicht automatisch lädt).

- Weitere Informationen über die Verwendung der DPAPI mit dem Computerspeicher (direkt) von einer ASP.NET-Webanwendung aus (ohne dass eine Enterprise Services-Anwendung benötigt wird) finden Sie unter "Vorgehensweise: Verwenden der DPAPI (Computerspeicher) von ASP.NET aus" im Abschnitt "Referenz" dieses Handbuchs.

Bei dem in dieser Vorgehensweise beschriebenen Ansatz wird eine .NET Serviced Component für die DPAPI-Verarbeitung verwendet, die in einer Enterprise Services- (COM+-) Serveranwendung ausgeführt wird; die Gründe hierfür werden im folgenden Abschnitt "Gründe für die Verwendung von Enterprise Services" näher erläutert. Darüber hinaus wird aus Gründen, die im Abschnitt "Gründe für die Verwendung eines Windows-Dienstes" näher ausgeführt werden, auch ein Windows-Dienst verwendet. Die Konfiguration der Lösung ist in Abbildung 1 dargestellt.

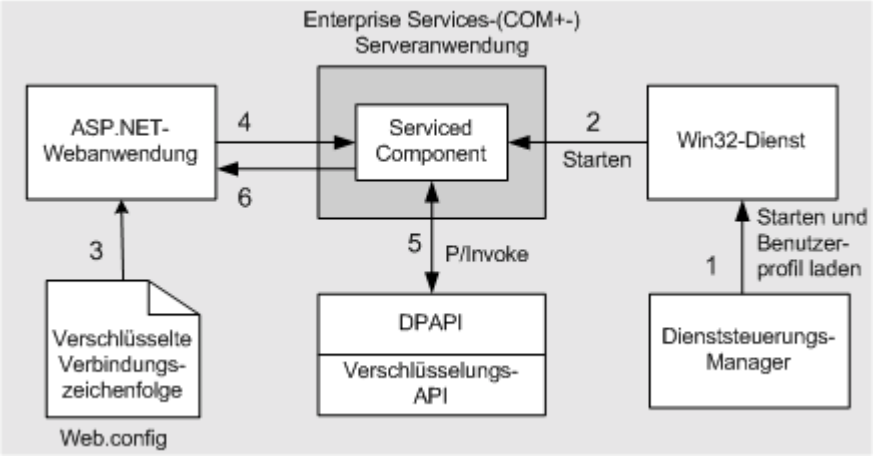

Abbildung 1
Die ASP.NET-Webanwendung nutzt eine Serviced Component in einer Enterprise Services-Serveranwendung für die Interaktion mit der DPAPI

Nachstehend die Abfolge der Ereignisse in Abbildung 1:

1. Der Dienststeuerungs-Manager von Windows startet den Win32-Dienst und lädt automatisch das Benutzerprofil, das mit dem Konto verbunden ist, unter dem der Dienst ausgeführt wird. Unter dem gleichen Windows-Konto wird auch die Enterprise Services-Anwendung ausgeführt.
2. Der Win32-Dienst ruft eine Startmethode für die Serviced Component auf, mit der die Enterprise Services-Anwendung gestartet und die Serviced Component geladen wird.
3. Die Webanwendung ruft die verschlüsselte Zeichenfolge aus der Datei **Web.config** ab.
4. Die Anwendung ruft eine Methode für die Serviced Component auf, um die Verbindungszeichenfolge zu entschlüsseln.
5. Die Serviced Component arbeitet mit der DPAPI zusammen, um unter Verwendung von **P/Invoke** die Win32-DPAPI-Funktionen aufzurufen.
6. Die entschlüsselte Zeichenfolge wird an die Webanwendung zurückgegeben.

Gründe für die Verwendung von Enterprise Services

Die DPAPI erfordert das Kennwort eines Windows-Kontos, um einen Verschlüsselungsschlüssel ableiten zu können. Das von der DPAPI verwendete Konto wird entweder aus dem aktuellen Threadtoken ermittelt (sofern der Thread, der die DPAPI aufruft, aktuell einen Identitätswechsel vornimmt), oder aus dem Prozesstoken. Darüber hinaus setzt die Verwendung der DPAPI mit dem Benutzerspeicher voraus, dass das mit dem Konto verbundene Benutzerprofil geladen wird. Hieraus ergibt sich für eine ASP.NET-Webanwendung, die die DPAPI mit dem Benutzerspeicher verwenden soll, die folgende Problemstellung:

- Aufrufe der DPAPI von einer ASP.NET-Anwendung, die unter dem standardmäßigen ASPNET-Konto ausgeführt wird, schlagen fehl, denn das ASPNET-Konto verfügt nicht über ein geladenes Benutzerprofil.
- Wenn eine ASP.NET-Webanwendung für einen Identitätswechsel ihrer Aufrufer konfiguriert ist, verfügt der Thread der ASP.NET-Anwendung über ein zugeordnetes Threadidentitätswechseltoken. Die Anmeldesitzung, die mit diesem Identitätswechseltoken verbunden ist, ist eine Netzwerkanmeldesitzung (die auf dem Server verwendet wird, um den Aufrufer zu repräsentieren). Bei Netzwerk-anmeldesitzungen werden keine Benutzerprofile geladen, und es ist auch nicht möglich, einen Verschlüsselungsschlüssel aus dem Kennwort abzuleiten, da der Server das Kennwort des Benutzers, dessen Identität gewechselt wurde, nicht kennt (es sei denn, die Anwendung arbeitet mit der Standardauthentifizierung).

Um diese Beschränkungen zu überwinden, können Sie eine Serviced Component in einer Enterprise Services-Serveranwendung (mit einer festen Prozessidentität) verwenden, um Ver- und Entschlüsselungsdienste unter Verwendung der DPAPI bereitzustellen.

Gründe für die Verwendung eines Windows-Dienstes

In dieser Lösung wird ein Windows-Dienst verwendet, um sicherzustellen, dass automatisch ein Benutzerprofil geladen wird. Wenn der Dienststeuerungs-Manager (Service Control Manager, SCM) von Windows einen Dienst startet, lädt dieser auch das Profil des Kontos, unter dem der Dienst gemäß seiner Konfiguration ausgeführt wird.

Der Dienst wird anschließend verwendet, um die Serviced Component zu laden, die bewirkt, dass die Enterprise Services-Serveranwendung (in einer Instanz von **Dllhost.exe**) gestartet wird.

Aufgrund der Tatsache, dass der Windows-Dienst und die Serviced Component beide für die Ausführung unter dem gleichen Konto mit minimalen Rechten konfiguriert sind, kann die Serviced Component auf das geladene Benutzerprofil zugreifen und dementsprechend die DPAPI-Funktionen für das Ver- und Entschlüsseln von Daten aufrufen.

Wenn die Serviced Component nicht von einem Windows-Dienst gestartet wird (d. h. wenn auf den Dienst verzichtet wurde), wird auch das Benutzerprofil nicht automatisch geladen. Es gibt zwar eine Win32-API, die aufgerufen werden kann, um ein Benutzerprofil zu laden (**LoadUserProfile**), diese setzt jedoch voraus, dass der aufrufende Code unter einem Konto aus der Gruppe der **Administratoren** ausgeführt wird, was wiederum dem Prinzip der Ausführung mit minimalen Rechten entgegenläuft.

Der Dienst muss immer ausgeführt werden, wenn die Methoden **Encrypt** und **Decrypt** der Serviced Component aufgerufen werden. Wenn Windows-Dienste gestoppt werden, wird das konfigurierte Profil automatisch entladen. An diesem Punkt stellen auch die DPAPI-Methoden der Serviced Component ihren Dienst ein.

Anforderungen

Im Folgenden finden Sie eine Liste der empfohlenen Hardware und Software und eine Beschreibung der Netzwerkinfrastruktur, der Fähigkeiten und Kenntnisse sowie der Service Packs, die Sie benötigen:

- Microsoft® SQL Server™ 2000 oder Microsoft Windows® XP als Betriebssystem
- Microsoft Visual Studio® .NET als Entwicklungssystem

Die in dieser Vorgehensweise erläuterten Verfahren setzen zudem Kenntnisse der ASP.NET-Webentwicklung mit dem Entwicklungstool Microsoft Visual C#™ voraus.

Zusammenfassung

Diese Vorgehensweise enthält folgende Verfahren:

1. Erstellen einer Serviced Component, die die Methoden **Encrypt** und **Decrypt** bereitstellt
2. Aufrufen der verwalteten DPAPI-Klassenbibliothek

3. Erstellen einer Blindklasse zum Starten der Serviced Component
4. Erstellen eines Windows-Kontos zum Ausführen der Enterprise Services-Anwendung und des Windows-Dienstes
5. Konfigurieren und Zuweisen eines starken Namens und Registrieren der Serviced Component
6. Erstellen einer Windows-Dienstanwendung zum Starten der Serviced Component
7. Installieren und Starten des Windows-Dienstes
8. Schreiben einer Webanwendung zum Testen der Ver- und Entschlüsselungsroutinen
9. Ändern der Webanwendung zum Lesen einer verschlüsselten Verbindungszeichenfolge aus der Konfigurationsdatei einer Anwendung

1. Erstellen einer Serviced Component, die die Methoden Encrypt und Decrypt bereitstellt

Mit dem nachstehenden Verfahren wird eine Serviced Component erstellt, die die Methoden **Encrypt** und **Decrypt** bereitstellt. In einem noch folgenden Verfahren werden diese Methoden von einer ASP.NET-Webanwendung aufgerufen, wenn diese Verschlüsselungsdienste benötigt.

▶ **So erstellen Sie eine Serviced Component, die die Methoden "Encrypt" und "Decrypt" bereitstellt**

1. Starten Sie Visual Studio .NET, und erstellen Sie ein neues C#-Klassenbibliotheksprojekt mit Namen **DPAPIComp**.
2. Benennen Sie **Class1.cs** im Projektmappen-Explorer in **DataProtectorComp.cs** um.
3. Benennen Sie in **DataProtetorComp.cs** die Klasse **Class1** in **DataProtectorComp** um, und weisen Sie auch dem Standardkonstruktor den entsprechenden Namen zu.
4. Fügen Sie einen Assemblyverweis auf die Assembly **System.EnterpriseServices.dll** hinzu.
5. Fügen Sie am Anfang von **DataProtectorComp.cs** die folgenden **using**-Anweisungen hinzu.

```
using System.EnterpriseServices;
using System.Security.Principal;
using System.Runtime.InteropServices;
```

6. Leiten Sie die **DataProtectorComp**-Klasse von der **ServicedComponent**-Klasse ab.

```
public class DataProtectorComp : ServicedComponent
```

7. Fügen Sie der **DataProtectorComp**-Klasse die folgenden leeren Methoden vom Typ **public** hinzu.

```
public byte[] Encrypt(byte[] plainText)
{}
public byte[] Decrypt(byte[] cipherText)
{}
```

2. Aufrufen der verwalteten DPAPI-Klassenbibliothek

Mit diesem Verfahren wird die verwaltete DPAPI-Klassenbibliothek zum Ver- und Entschlüsseln von Daten aufgerufen. Diese Klassenbibliothek kapselt die Aufrufe der Win32-DPAPI-Funktionen ein. Wenn Sie diese Klassenbibliothek noch nicht erstellt haben, lesen Sie "Vorgehensweise: Erstellen einer DPAPI-Bibliothek" im Abschnitt "Referenz" dieses Handbuchs, und führen Sie die hier genannten Schritte durch.

▶ **So rufen Sie die verwaltete DPAPI-Klassenbibliothek auf**

1. Fügen Sie einen Dateiverweis auf die Assembly **DataProtection.dll** hinzu.
2. Fügen Sie in **DataProtectorComp.cs** unterhalb der vorhandenen **using**-Anweisungen die folgende **using**-Anweisung hinzu.

```
using DataProtection;
```

3. Fügen Sie der **Encrypt**-Methode den folgenden Code hinzu, um die übergebenen Daten zu verschlüsseln.

```
DataProtector dp = new DataProtector( DataProtector.Store.USE_USER_STORE );
byte[] cipherText = null;
try
{
  cipherText = dp.Encrypt(plainText, null);
}
catch(Exception ex)
{
  throw new Exception("Exception encrypting. " + ex.Message);
}
return cipherText;
```

4. Fügen Sie der **Decrypt**-Methode den folgenden Code hinzu, um den übergebenen verschlüsselten Text zu entschlüsseln.

```
DataProtector dp = new DataProtector( DataProtector.Store.USE_USER_STORE );
byte[] plainText = null;

try
{
  plainText = dp.Decrypt(cipherText,null);
}
catch(Exception ex)
{
  throw new Exception("Exception decrypting. " + ex.Message);
}
return plainText;
```

3. Erstellen einer Blindklasse zum Starten der Serviced Component

In diesem Verfahren wird eine Blindklasse erstellt, die eine einzige **Launch**-Methode bereitstellt. Diese wird vom Windows-Dienst aufgerufen, um die Enterprise Services-Anwendung zu starten, die als Host für die Serviced Component fungiert.

▶ **So erstellen Sie eine Blindklasse zum Starten der Serviced Component**

1. Fügen Sie dem Projekt eine neue C#-Klasse mit Namen **Launcher.cs** hinzu.
2. Fügen Sie der Klasse die folgende Methode hinzu. Diese Methode wird vom Dienst aufgerufen, wenn der Dienst gestartet wird.

```
public bool Launch()
{
  return true;
}
```

3. Klicken Sie im Menü **Erstellen** auf **Projektmappe erstellen**.

4. Erstellen eines Windows-Kontos zum Ausführen der Enterprise Services-Anwendung und des Windows-Dienstes

In diesem Verfahren wird ein Windows-Konto erstellt, das zum Ausführen der Enterprise Services-Anwendung verwendet wird, die als Host für die Serviced Component **DataProtectorComp** und des Windows-Dienstes fungiert. Darüber hinaus wird ein Benutzerprofil für das neue Konto erstellt. Dieses Benutzerprofil wird von der DPAPI benötigt, wenn diese auf den Benutzerspeicher zugreift.

▶ **So erstellen Sie ein Windows-Konto zum Ausführen der Enterprise Services-Anwendung und des Windows-Dienstes**

1. Erstellen Sie ein neues lokales Benutzerkonto mit Namen **DPAPIAccount**. Geben Sie ein Kennwort ein, deaktivieren Sie das Kontrollkästchen **Benutzer muss Kennwort bei der nächsten Anmeldung ändern**, und aktivieren Sie das Kontrollkästchen **Kennwort läuft nie ab**.
2. Erteilen Sie dem Konto unter Verwendung des Tools **Lokale Sicherheitsrichtlinie** in der Programmgruppe **Verwaltung** die Rechte **Lokal anmelden** und **Anmelden als Stapelverarbeitungsauftrag**.

▶ **So erstellen Sie ein Benutzerprofil für das neue Konto**

1. Melden Sie sich von Windows ab.
2. Melden Sie sich unter Verwendung des neuen Kontos **DPAPIAccount** wieder an.
   Hiermit wird ein Benutzerprofil für dieses Konto erstellt.
3. Melden Sie sich von Windows ab, und melden Sie sich unter Ihrem normalen Entwicklerkonto wieder an.

5. Konfigurieren, Zuweisen eines starken Namens und Registrieren der Serviced Component

In diesem Verfahren wird die Serviced Component-Assembly signiert, um ihr einen starken Namen zuzuweisen. Für Assemblys, die Serviced Components enthalten, ist dies zwingend erforderlich. Anschließend fügen Sie der Serviced Component-Assembly Assemblyebenenattribute hinzu, die für die Konfiguration der Serviced Component im COM+-Katalog verwendet werden. Hiernach verwenden Sie das Dienstprogramm **Regsvcs.exe** zum Registrieren der Serviced Component und erstellen eine COM+-Serveranwendung, die als Host fungiert. Abschließend legen Sie die "Ausführen als"-Identität der COM+-Anwendung auf das Dienstkonto fest, das im vorherigen Verfahren erstellt wurde.

▶ **So können Sie die Serviced Component konfigurieren, ihr einen starken Namen zuweisen und sie registrieren**

1. Öffnen Sie ein Befehlsfenster, und wechseln Sie in den Projektordner **DPAPIComp**.
2. Erzeugen Sie mithilfe des Dienstprogramms **sn.exe** ein Schlüsselpaar zum Signieren der Assembly.

```
sn -k dpapicomp.snk
```

3. Kehren Sie zu Visual Studio .NET zurück, und öffnen Sie **Assemblyinfo.cs**.

4. Suchen Sie das **AssemblyKeyFile**-Attribut, und fügen Sie den Pfad zur Schlüsseldatei im Projektordner hinzu.

```
[assembly: AssemblyKeyFile(@"..\..\dpapicomp.snk")]
```

5. Fügen Sie am Anfang der Datei die folgende **using**-Anweisung hinzu.

```
using System.EnterpriseServices;
```

6. Fügen Sie die folgenden Assemblyebenenattribute zum Konfigurieren der COM+-Anwendung als Serveranwendung und zur Angabe des Namens der Anwendung hinzu.

```
[assembly: ApplicationActivation(ActivationOption.Server)]
[assembly: ApplicationName("DPAPI Helper Application")]
```

7. Klicken Sie im Menü **Erstellen** auf **Projektmappe erstellen**, um das Serviced Component-Projekt zu erstellen.
8. Öffnen Sie ein Befehlsfenster, und wechseln Sie ins Ausgabeverzeichnis des Projekts, das die Datei **DPAPIComp.dll** enthält.
9. Registrieren Sie die Serviced Component mithilfe von **Regsvcs.exe**, und erstellen Sie die COM+-Anwendung.

```
regsvcs DPAPIComp.dll
```

10. Starten Sie das MMC-Snap-In (Microsoft Management Console) für Komponentendienste.
11. Erweitern Sie die Ordner **Komponentendienste**, **Computer**, **Arbeitsplatz** und **COM+-Anwendungen**.
12. Suchen Sie **DPAPI Helper Application**, klicken Sie mit der rechten Maustaste auf den Eintrag, und klicken Sie dann auf **Eigenschaften**.
13. Klicken Sie auf die Registerkarte **Aktivierung**, und vergewissern Sie sich, dass als Anwendungstyp **Serveranwendung** festgelegt wurde.
14. Klicken Sie auf die Registerkarte **Identität**, und klicken Sie dann auf das Optionsfeld **Dieser Benutzer**.
15. Geben Sie **DPAPIAccount** als Benutzer sowie das zugehörige Kennwort ein, und klicken Sie dann auf **OK**, um das Dialogfeld **Eigenschaften** zu schließen.

6. Erstellen einer Windows-Dienstanwendung zum Starten der Serviced Component

Mit diesem Verfahren wird eine einfache Windows-Dienstanwendung erstellt, die beim Start die Serviced Component startet. Hiermit wird sichergestellt, dass das Profil des konfigurierten Kontos geladen wird und dass die Serviced Component zum Ver- und Entschlüsseln von Daten zur Verfügung steht.

▶ **So erstellen Sie eine Windows-Dienstanwendung zum Starten der Serviced Component**

1. Öffnen Sie eine neue Instanz von Visual Studio .NET, und erstellen Sie ein neues C#-Windows-Dienstprojekt mit Namen **DPAPIService**.
2. Benennen Sie **Service1.cs** im Projektmappen-Explorer in **DPAPIService.cs** um.
3. Benennen Sie in **DPAPIService.cs** den Dienst **Service1** in **DPAPIService** um, und weisen Sie auch dem Standardkonstruktor den entsprechenden Namen zu.
4. Suchen Sie in **DPAPIService.cs** die **InitializedComponent**-Methode, und ändern Sie den Dienstnamen in **DPAPIService**.

5. Fügen Sie Verweise auf die Assemblys **System.EnterpriseServices.dll** und **System.Configuration.Install.dll** hinzu.
6. Fügen Sie einen Dateiverweis auf die Assembly **DPAPIComp** hinzu.
7. Fügen Sie am Anfang von **DPAPIService.cs** unterhalb der vorhandenen **using**-Anweisungen die folgende **using**-Anweisung hinzu.

```
using DPAPIComp;
```

8. Suchen Sie die **Main**-Methode, und ersetzen Sie den folgenden Code

```
ServicesToRun = new System.ServiceProcess.ServiceBase[]{new Service1()};
```

durch diese Zeile:

```
ServicesToRun = new System.ServiceProcess.ServiceBase[]{new DPAPIService()};
```

9. Suchen Sie die **OnStart**-Methode, und fügen Sie den folgenden Code hinzu, mit dem die **DPAPIComp**-Komponente bei jedem Start des Dienstes gestartet wird.

```
Launcher launchComponent = new Launcher();
launchComponent.Launch();
```

10. Fügen Sie dem Projekt eine neue C#-Klassendatei mit Namen **DPAPIServiceInstaller** hinzu.
11. Fügen Sie am Anfang von **DPAPIServiceInstaller** unterhalb der vorhandenen **using**-Anweisung die folgende **using**-Anweisungen hinzu.

```
using System.ComponentModel;
using System.ServiceProcess;
using System.Configuration.Install;
```

12. Leiten Sie die **DPAPIServiceInstaller**-Klasse von der **Installer**-Klasse ab.

```
public class DPAPIServiceInstaller : Installer
```

13. Fügen Sie das **RunInstaller**-Attribut auf Klassenebene wie folgt hinzu:

```
[RunInstaller(true)]
public class DPAPIServiceInstaller : Installer
```

14. Fügen Sie der **DPAPIServiceInstaller**-Klasse die beiden folgenden Membervariablen vom Typ **private** hinzu. Diese Objekte werden beim Installieren des Dienstes verwendet.

```
private ServiceInstaller dpApiInstaller;
private ServiceProcessInstaller dpApiProcessInstaller;
```

15. Fügen Sie dem Konstruktor der **DPAPIServiceInstaller**-Klasse den folgenden Code hinzu.

```
dpApiInstaller = new ServiceInstaller();
dpApiInstaller.StartType = System.ServiceProcess.ServiceStartMode.Manual;
dpApiInstaller.ServiceName = "DPAPIService";
dpApiInstaller.DisplayName = "DPAPI Service";
```

```
Installers.Add (dpApiInstaller);
dpApiProcessInstaller = new ServiceProcessInstaller();
dpApiProcessInstaller.Account = ServiceAccount.User;
Installers.Add (dpApiProcessInstaller);
```

16. Klicken Sie im Menü **Erstellen** auf **Projektmappe erstellen**.

7. Installieren und Starten der Windows-Dienstanwendung

Mit diesem Verfahren wird der Windows-Dienst unter Verwendung des Dienstprogramms **installutil.exe** installiert und anschließend gestartet.

▶ **So installieren und starten Sie die Windows-Dienstanwendung**

1. Öffnen Sie ein Befehlsfenster, und wechseln Sie in das Verzeichnis **Bin\Debug** im Projektordner **DPAPIService**.
2. Führen Sie das Dienstprogramm **installutil.exe** aus, um den Dienst zu installieren.

```
Installutil.exe DPAPIService.exe
```

3. Geben Sie im Dialogfeld **Dienstanmeldung festlegen** den Benutzernamen und das Kennwort des Kontos ein, das in Verfahren 4, "Erstellen eines Windows-Kontos zum Ausführen der Enterprise Services-Anwendung und des Windows-Dienstes", erstellt wurde, und klicken Sie dann auf **OK**.

   Der Benutzername muss das Format "Autorität\Benutzername" aufweisen.

   Zeigen Sie die Ausgabe des Dienstprogramms **installutil.exe** an, und vergewissern Sie sich, dass der Dienst ordnungsgemäß installiert wurde.
4. Starten Sie das MMC-Snap-In Dienste aus der Programmgruppe **Verwaltung**.
5. Starten Sie den DPAPI-Dienst.

8. Schreiben einer Webanwendung zum Testen der Ver- und Entschlüsselungsroutinen

Mit diesem Verfahren wird eine einfache Webanwendung entwickelt, die Sie zum Testen der Ver- und Entschlüsselungsroutinen verwenden können. Später werden Sie die Anwendung auch verwenden, um verschlüsselte Daten in der Datei **Web.config** zu entschlüsseln.

▶ **So schreiben Sie eine Webanwendung zum Testen der Ver- und Entschlüsselungsroutinen**

1. Fügen Sie der vorhandenen Lösung **DPAPIComp** ein neues C#-Webanwendungs-projekt mit Namen **DPAPIWeb** hinzu.
2. Fügen Sie einen Assemblyverweis auf **System.EnterpriseServices** und einen Projektverweis auf das Projekt **DPAPIComp** hinzu.
3. Öffnen Sie **WebForm1.aspx** im Entwurfsmodus, und erstellen Sie ein Formular ähnlich dem in Abbildung 1 gezeigten. Verwenden Sie die in Tabelle 1 aufgeführten IDs für die einzelnen Steuerelemente.

   Tabelle 1: Steuerelement-IDs für **WebForm1.aspx**

Steuerelement	ID
Textfeld **Data To Encrypt**	txtDataToEncrypt
Textfeld **Encrypted Data**	txtEncryptedData
Textfeld **Decrypted Data**	txtDecryptedData
Schaltfläche **Encrypt**	btnEncrypt
Schaltfläche **Decrypt**	btnDecrypt
Bezeichnungsfeld **Error**	lblError

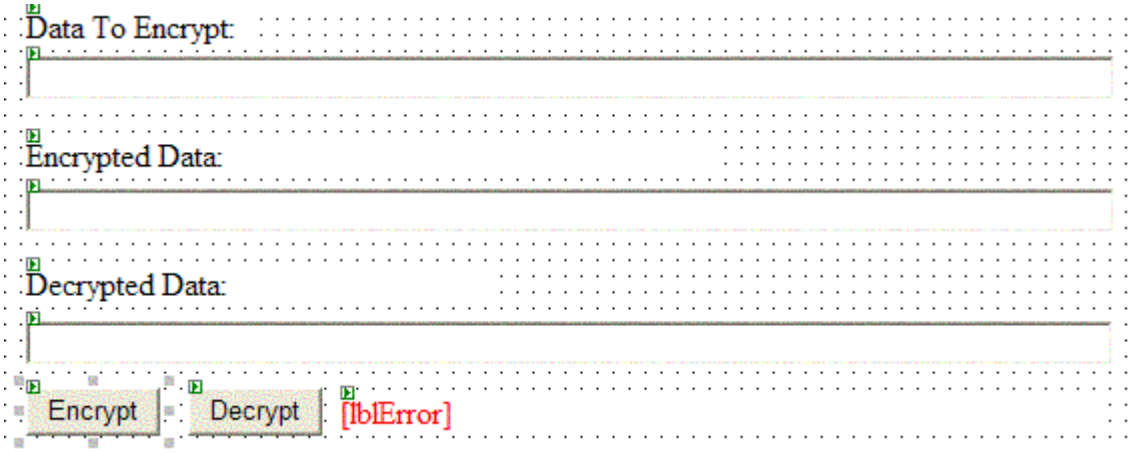

Abbildung 1
Webformular DPAPIWeb

4. Doppelklicken Sie auf die Schaltfläche **Encrypt**, um den Ereignishandler für das Klickereignis der Schaltfläche anzuzeigen.
5. Fügen Sie am Anfang der Datei unterhalb der vorhandenen **using**-Anweisungen die folgende **using**-Anweisungen hinzu.

```
using System.Text;
using DPAPIComp;
```

6. Kehren Sie zum Ereignishandler für das Klickereignis der Schaltfläche **Encrypt** zurück, und fügen Sie den folgenden Code hinzu, um die Serviced Component **DataProtectorComp** zum Verschlüsseln der Daten aufzurufen, die über das Webformular eingegeben werden.

```
DataProtectorComp dp = new DataProtectorComp();
try
{
  byte[] dataToEncrypt = Encoding.ASCII.GetBytes(txtDataToEncrypt.Text);
  txtEncryptedData.Text = Convert.ToBase64String(
                                    dp.Encrypt(dataToEncrypt));
}
catch(Exception ex)
{
  lblError.ForeColor = Color.Red;
  lblError.Text = "Exception.<br>" + ex.Message;
  return;
}
lblError.Text = "";
```

7. Zeigen Sie das Webformular erneut an, und doppelklicken Sie auf die Schaltfläche **Decrypt**, um einen Ereignishandler für das Klickereignis der Schaltfläche zu erstellen.
8. Fügen Sie folgenden Code hinzu, um die Serviced Component **DataProtectorComp** zum Entschlüsseln der vorher verschlüsselten Daten aufzurufen, die sich nun im Feld **txtEncryptedData** befinden.

```
DataProtectorComp dp = new DataProtectorComp();
try
{
  byte[] dataToDecrypt = Convert.FromBase64String(txtEncryptedData.Text);
  txtDecryptedData.Text = Encoding.ASCII.GetString(
```

```
                                    dp.Decrypt(dataToDecrypt));
    }
    catch(Exception ex)
    {
      lblError.ForeColor = Color.Red;
      lblError.Text = "Exception.<br>" + ex.Message;
      return;
    }
    lblError.Text = "";
```

9. Klicken Sie im Menü **Erstellen** auf **Projektmappe erstellen**.
10. Klicken Sie im Projektmappen-Explorer mit der rechten Maustaste auf **WebForm1.aspx**, und klicken Sie dann auf **In Browser anzeigen**.
11. Geben Sie eine Textzeichenfolge in das Feld **Data to Encrypt** ein.
12. Klicken Sie auf die Schaltfläche **Encrypt**. Nun wird die Serviced Component **DataProtector** der COM+-Anwendung aufgerufen. Die verschlüsselten Daten sollten im Feld **Encrypted Data** angezeigt werden.
13. Klicken Sie auf die Schaltfläche **Decrypt**, und vergewissern Sie sich, dass die ursprüngliche Textzeichenfolge im Feld **Decrypted Data** angezeigt wird.
14. Schließen Sie das Browserfenster.

Hinweis: Wenn eine Fehlermeldung "Zugriff verweigert" angezeigt wird, die angibt, dass die ProgID der Komponente nicht aus HKEY_CLASSES_ROOT gelesen werden kann, müssen Sie **Regsvcs.exe** möglicherweise noch einmal ausführen, um die Serviced Component erneut zu registrieren.

Diese Fehlermeldung wird angezeigt, wenn Sie die Assembly mit der Serviced Component erneut kompiliert, die Assembly jedoch nicht erneut registriert haben. Da sich die Assemblyversion mit jedem neuen Build ändert (aufgrund des standardmäßigen Assemblyversionsattributs "1.0.*"), wird bei jedem nachfolgenden Build eine neue CLSID erzeugt. Der Fehler wird aufgrund der Tatsache ausgegeben, dass ASP.NET nicht auf diese CLSID in der Registrierung zugreifen kann, da diese noch nicht existiert. Führen Sie **Regsvcs.exe** erneut aus, und starten Sie die Webanwendung neu, um das Problem zu beheben.

---

9. Ändern der Webanwendung zum Lesen einer verschlüsselten Verbindungszeichenfolge aus der Konfigurationsdatei einer Anwendung

Mit diesem Verfahren wird eine verschlüsselte Datenbank-Verbindungszeichenfolge in einem <**appSettings**>-Element in der Datei **Web.config** der Anwendung platziert. Anschließend fügen Sie Code zum Lesen und Entschlüsseln dieser Zeichenfolge aus der Konfigurationsdatei hinzu.

▶ **So ändern Sie die Webanwendung zum Lesen einer verschlüsselten Verbindungszeichenfolge aus einer Anwendungskonfigurationsdatei**

1. Kehren Sie zu Visual Studio .NET zurück, und zeigen Sie **WebForm1.aspx** im Entwurfsmodus an.
2. Fügen Sie dem Formular eine weitere Schaltfläche hinzu. Legen Sie die **Text**-Eigenschaft auf **Decrypt string from config file** und die **ID**-Eigenschaft auf **btnDecryptConfig** fest.
3. Doppelklicken Sie auf die Schaltfläche, um einen Ereignishandler für das Klickereignis der Schaltfläche zu erstellen.
4. Fügen Sie am Anfang der Datei unterhalb der vorhandenen **using**-Anweisungen die folgende **using**-Anweisung hinzu.

```
using System.Configuration;
```

5. Kehren Sie zum Ereignishandler **btnDecryptConfig_Click** zurück, und fügen Sie den folgenden Code hinzu, um eine Datenbank-Verbindungszeichenfolge aus dem Abschnitt **<appSettings>** der Datei **Web.config** abzurufen.

```
DataProtectorComp dec = new DataProtectorComp();
try
{
  string appSettingValue =
            ConfigurationSettings.AppSettings["connectionString"];
  byte[] dataToDecrypt = Convert.FromBase64String(appSettingValue);
  string connStr = Encoding.ASCII.GetString(
                              dec.Decrypt(dataToDecrypt));
  txtDecryptedData.Text = connStr;
}
catch(Exception ex)
{
  lblError.ForeColor = Color.Red;
  lblError.Text = "Exception.<br>" + ex.Message;
  return;
}
lblError.Text = "";
```

6. Klicken Sie im Menü **Erstellen** auf **Projektmappe erstellen**, um die Projekte neu zu erstellen.
7. Klicken Sie im Projektmappen-Explorer mit der rechten Maustaste auf **WebForm1.aspx**, und klicken Sie dann auf **In Browser anzeigen**.
8. Geben Sie eine Datenbank-Verbindungszeichenfolge wie die folgende in das Feld **Data to Encrypt** ein.

```
server=(local);Integrated Security=SSPI; database=Northwind
```

9. Klicken Sie auf die Schaltfläche **Encrypt**.
10. Markieren Sie den verschlüsselten Chiffrierungstext, und kopieren Sie ihn in die Zwischenablage.
11. Wechseln Sie zu Visual Studio.NET, öffnen Sie **Web.config**, und fügen Sie außerhalb des **<system.web>**-Elements das folgende **<appSettings>**-Element hinzu. Weisen Sie die verschlüsselte Verbindungszeichenfolge, die sich gegenwärtig in der Zwischenablage befindet, dem **value**-Attribut zu.

```
<appSettings>
   <add key="connectionString" value="encrypted connection string" />
</appSettings>
```

12. Speichern Sie die Datei **Web.config**.
13. Klicken Sie auf die Schaltfläche **Decrypt string from config file**, und vergewissern Sie sich, dass die verschlüsselte Datenbank-Verbindungszeichenfolge erfolgreich aus der Datei **Web.config** ausgelesen werden konnte und im Feld **Decrypted data** angezeigt wird.

Referenzen

- "Windows Data Protection" in MSDN (http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnsecure/html/windataprotection-dpapi.asp, englischsprachig)
- "Vorgehensweise: Erstellen einer DPAPI-Bibliothek" im Abschnitt "Referenz" dieses Handbuchs
- "Vorgehensweise: Verwenden von DPAPI (Computerspeicher) von ASP.NET aus" im Abschnitt "Referenz" dieses Handbuchs