# Kapitel 9 – Sicherheit bei Enterprise Services

## Erstellen sicherer ASP.NET-Anwendungen

### Authentifizierung, Autorisierung und sichere Kommunikation

J.D. Meier, Alex Mackman, Michael Dunner und Srinath Vasireddy

Microsoft Corporation

Oktober 2002

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

**Zusammenfassung**

In diesem Kapitel wird erläutert, wie die Geschäftsfunktionalität in Service Components innerhalb von Enterprise Services-Anwendungen gesichert wird. Es wird gezeigt, wie und wann die Enterprise Services (COM+)-Rollen für die Autorisierung verwendet und wie die RPC-Authentifizierung und der Identitätswechsel konfiguriert werden. Des Weiteren wird beschrieben, wie Serviced Components über eine ASP.NET-Webanwendung sicher aufgerufen und wie der Sicherheitskontext des ursprünglichen Aufrufers über eine Serviced Component der mittleren Ebene identifiziert und übermittelt wird.

**Inhalt**

Den .NET-Komponenten stehen traditionelle COM+-Dienste zur Verfügung, z. B. die verteilten Transaktionen, Just-In-Time-Aktivierungen, das Objektpooling und die Parallelitätsverwaltung. Bei .NET werden diese Dienste als Enterprise Services bezeichnet. Diese sind für viele .NET-Komponenten der mittleren Ebene von entscheidender Bedeutung, die in .NET-Webanwendungen ausgeführt werden.

Wenn Sie zu einer .NET-Komponente Dienste hinzufügen möchten, müssen Sie die Komponentenklasse von der Basisklasse **EnterpriseServices**.**ServicedComponent** ableiten und dann exakte Dienstanforderungen angeben, wobei Sie .NET-Attribute verwenden, die in die Assembly kompiliert werden, die den Host für die Komponente darstellt.

In diesem Kapitel wird beschrieben, wie sichere Serviced Components erstellt und diese über ASP.NET-Webanwendungen aufgerufen werden.

# Sicherheitsarchitektur

Die von den Enterprise Services-Anwendungen unterstützten Features für die Authentifizierung, Autorisierung und sichere Kommunikation werden in Abbildung 9.1 dargestellt. Die in Abbildung 9.1 gezeigte Clientanwendung ist eine ASP.NET-Webanwendung.

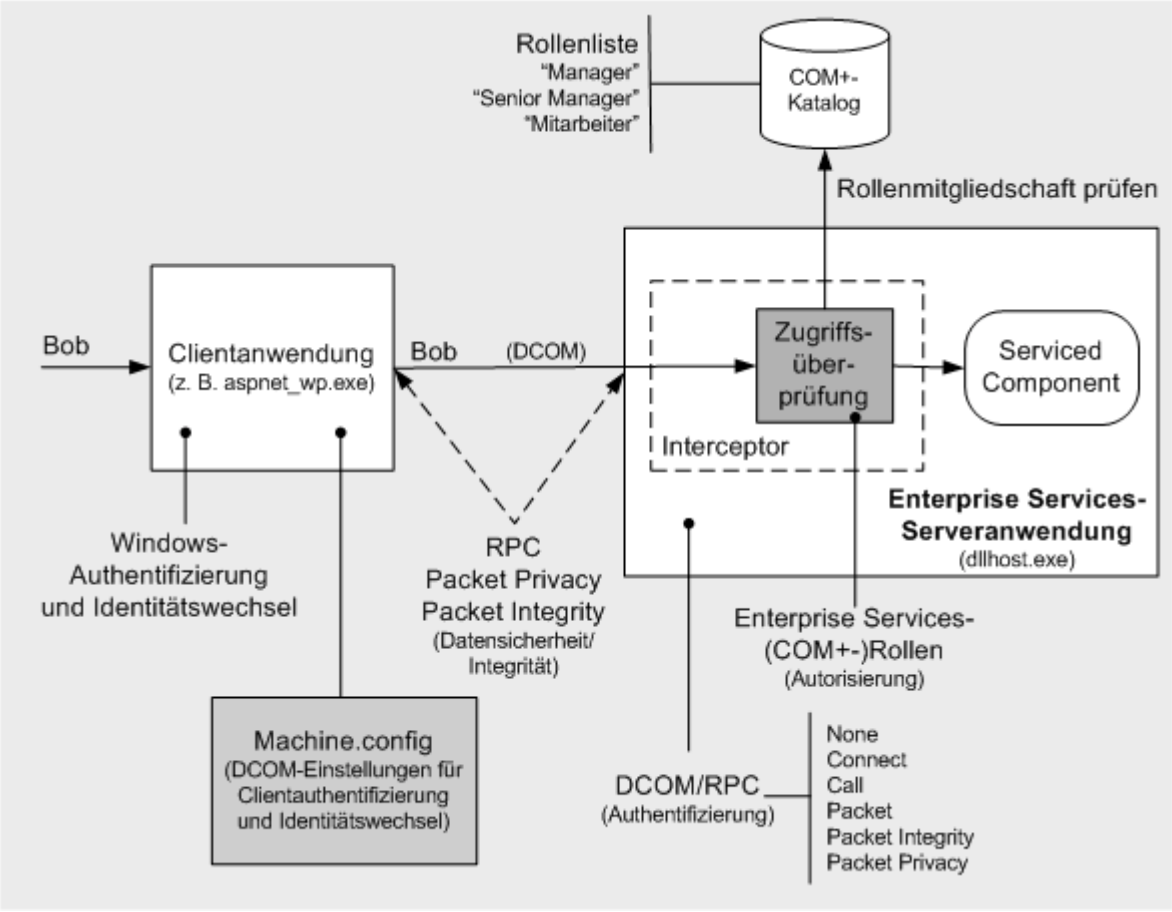


Abbildung 9.1
*Auf Enterprise Services basierende Sicherheitsarchitektur*

Beachten Sie, das die Funktionen für die Authentifizierung und sichere Kommunikation vom zugrunde liegenden RPC-Transport bereitgestellt werden, der von DCOM (Distributed COM) verwendet wird. Die Autorisierung wird von Enterprise Services (COM+)-Rollen bereitgestellt.

Nachfolgend sind die Hauptelemente der Enterprise Services-Sicherheitsarchitektur zusammengefasst:

- Enterprise Services-Anwendungen verwenden die RPC-Authentifizierung zum Authentifizieren von Aufrufern. Das bedeutet, dass der Aufrufer über die Kerberos- oder NTLM-Authentifizierung authentifiziert wird, wenn Sie nicht bestimmte Schritte zum Deaktivieren der Authentifizierung durchführen.
- Die Autorisierung wird über Enterprise Services (COM+)-Rollen bereitgestellt, die Gruppen- oder Benutzerkonten des Betriebssystems Microsoft® Windows® enthalten können. Die Rollenmitgliedschaft ist im COM+-Katalog definiert und wird mit dem Tool für Komponentendienste verwaltet.

---

**Hinweis:** Wenn die Enterprise Services-Anwendung den Identitätswechsel verwendet, steht die Aufruferautorisierung über Windows-Zugriffssteuerungslisten für gesicherte Ressourcen ebenfalls zur Verfügung.

---

- Wenn ein Client (z. B. eine ASP.NET-Webanwendung) eine Methode für eine Services Component aufruft, greift die Enterprise Services-Interceptionsebene im Anschluss an den Authentifizierungsprozess auf den COM+-Katalog zu, um die Rollenmitgliedschaft des Clients zu ermitteln. Dann wird überprüft, ob die Mitgliedschaft der Rolle(n) den autorisierten Zugriff auf die aktuelle Anwendung, Komponente, Schnittstelle und Methode zulässt.
- Wenn die Rollenmitgliedschaft des Clients den Zugriff gewährt, wird die Methode aufgerufen. Wenn der Client keiner geeigneten Rolle angehört, wird der Aufruf zurückgewiesen und optional ein Sicherheitsereignis generiert, um den fehlgeschlagenen Zugriffsversuch widerzuspiegeln.

---

**Wichtig:** Um eine sinnvolle rollenbasierte Autorisierung in einer Enterprise Services-Anwendung zu implementieren, die von einer ASP.NET-Webanwendung aufgerufen wird, müssen die Windows-Authentifizierung und der Identitätswechsel innerhalb der ASP.NET-Webanwendung verwendet werden, um sicherzustellen, dass der Sicherheitskontext des ursprünglichen Aufrufers die Serviced Component durchläuft.

---

- Zum Sichern der DCOM-Kommunikationsverbindung zwischen Client- und Serveranwendungen kann entweder die Authentifizierungsebene der RPC-Paketintegrität (um die Nachrichtenintegrität bereitzustellen) oder der RPC-Paketsicherheit (um die Nachrichtenvertraulichkeit bereitzustellen) verwendet werden.

## Gatekeeper und Gates

Das Enterprise Services-Laufzeitmodul fungiert als Gatekeeper für Serviced Components. Die einzelnen Gates (Autorisierungspunkte) in einer Enterprise Services-Anwendung werden in Abbildung 9.2 gezeigt. Sie können diese Gates unter Verwendung von Enterprise Services-Rollen konfigurieren, die Sie mit geeigneten Gruppen- und Benutzerkonten von Windows auffüllen müssen.

---

**Hinweis:** Sie müssen ebenfalls sicherstellen, dass die Zugriffsüberprüfung (rollenbasierte Sicherheit) für Ihre Enterprise Services-Anwendung aktiviert ist und die entsprechende Authentifizierungsebene verwendet wird. Weitere Informationen zum Konfigurieren der Sicherheit finden Sie unter "Konfigurieren der Sicherheit" weiter unten in diesem Kapitel.

---

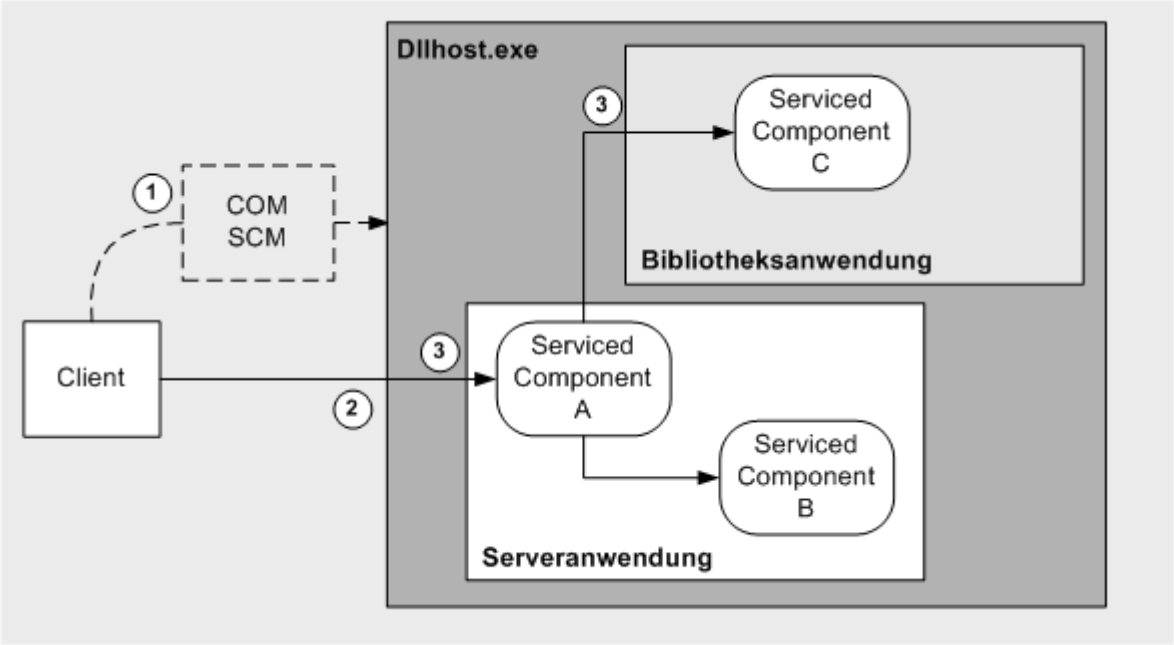


Abbildung 9.2
*Gatekeeper in einer Enterprise Services-Anwendung*

Als Reaktion auf einen Client, der einen Methodenaufruf für eine Serviced Component startet, werden drei unterschiedliche Zugriffsüberprüfungen durchgeführt. Diese werden in Abbildung 9.2 veranschaulicht und nachfolgend beschrieben:

1. Wenn ein Aufruf für eine Serviced Component zu einer Aktivierungsanforderung führt (und zur Erstellung einer neuen Instanz des COM+-Ersatzprozesses, **Dllhost.exe**) wird die erste Zugriffsüberprüfung von dem Untersystem durchgeführt, das zum Aktivieren von Enterprise Services-Anwendungen verantwortlich ist, dem COM-Dienststeuerungs-Manager (SCM, Service Control Manager).

   Der Aufrufer muss Mitglied mindestens einer Rolle sein, die innerhalb der Anwendung definiert ist, um diese Zugriffsüberprüfung erfolgreich zu bestehen.
2. Wenn der Aufruf des Clients die Prozessinstanz **Dllhost.exe** erreicht, wird eine zweite Zugriffsüberprüfung durchgeführt.

   Wenn der Aufrufer wiederum Mitglied mindestens einer Rolle ist, die innerhalb der Anwendung definiert ist, wird diese Überprüfung erfolgreich abgeschlossen.
3. Die abschließende Zugriffsüberprüfung erfolgt, wenn der Aufruf des Clients entweder eine Server- oder Bibliotheksanwendung erreicht.

   Damit diese Zugriffsüberprüfung erfolgreich durchgeführt wird, muss der Aufrufer Mitglied einer Rolle sein, die entweder der Schnittstelle, Klasse oder Methode zugeordnet ist, die das Aufrufsziel des Clients darstellt.

---

**Wichtig:** Nachdem ein Aufruf eine Methode für eine Serviced Component startet, erfolgen keine weiteren Zugriffsüberprüfungen, wenn die Komponente mit anderen Komponenten kommuniziert, die sich in derselben Anwendung befinden. Es werden jedoch Zugriffsüberprüfungen durchgeführt, wenn eine Komponente eine andere Komponente in einer anderen Anwendung (Bibliotheks- oder Serveranwendung) aufruft.

---

# Verwenden von Serveranwendungen zum Erhöhen der Sicherheit

Sie sollten eine Serveranwendung verwenden, wenn Ihre Anwendung eine Authentifizierungsebene durchsetzen muss, da sie z. B. die Verschlüsselung erfordert, um sicherzustellen, dass die an eine Serviced Component gesendeten Daten beim Transport im Netzwerk geheim bleiben und nicht manipuliert werden können.

Die Authentifizierungsebene kann für Serveranwendungen erzwungen werden, während Bibliotheksanwendungen die Authentifizierungsebene vom Hostprozess erben.

Verwenden Sie das **ApplicationActivation**-Attribut der Assemblyebene, wie unten gezeigt, um den Aktivierungstyp einer Enterprise Services-Anwendung zu konfigurieren.

```
[assembly: ApplicationActivation(ActivationOption.Server)]
```

Dies ist mit dem Einstellen der Option **Aktivierungstyp** auf den Wert **Serveranwendung** vergleichbar (auf der Seite **Aktivierung** des Dialogfelds **Eigenschaften** der Anwendung innerhalb der Komponentendienste).

# Sicherheit für Server- und Bibliotheksanwendungen

Die rollenbasierte Sicherheit funktioniert für prozessinterne Bibliotheksanwendungen und prozessexterne Serveranwendungen auf ähnliche Weise.

Beachten Sie die folgenden Unterschiede für Bibliotheksanwendungen:

- **Rechte** – Die Rechte einer Bibliotheksanwendung werden durch die Rechte des Clientprozesses (Hostprozesses) bestimmt. Wenn der Clientprozess z. B. mit Administratorrechten ausgeführt wird, verfügt die Bibliotheksanwendung ebenfalls über die Administratorrechte.
- **Identitätswechsel** – Die Identitätswechselebene einer Bibliotheksanwendung wird vom Clientprozess geerbt und kann nicht explizit festgelegt werden.
- **Authentifizierung** – Die Authentifizierungsebene einer Bibliotheksanwendung wird vom Clientprozess geerbt. Bei Bibliotheksanwendungen können Sie die Authentifizierung explizit aktivieren oder deaktivieren. Diese Option steht auf der Seite **Sicherheit** im Dialogfeld **Eigenschaften** einer Bibliotheksanwendung zur Verfügung.

  Diese Option wird normalerweise verwendet, um nicht authentifizierte Rückrufe von anderen prozessexternen COM-Komponenten zu unterstützen.

## Zuordnen von Rollen zu Klassen, Schnittstellen oder Methoden

Bei der Verwendung von Bibliotheksanwendungen sollten Sie immer Rollen auf der Klassen-, Schnittstellen- oder Methodenebene zuordnen. Dies gilt auch für Serveranwendungen.

Benutzer, die innerhalb von Bibliotheksanwendungsrollen definiert sind, können nicht zum Sicherheitsdeskriptor des Clientprozesses hinzugefügt werden. Das bedeutet, dass Sie zumindest die Sicherheit auf Klassenebene verwenden müssen, um es einer Bibliotheksanwendung zu ermöglichen, eine rollenbasierte Autorisierung durchzuführen.

# Anforderungen an die Codezugriffssicherheit

Die Codezugriffssicherheit (CAS, Code Access Security) erfordert, dass ein Code bestimmte Berechtigungen besitzt, damit er bestimmte Operationen durchführen und auf eingeschränkte Ressourcen zugreifen kann. Die Codezugriffssicherheit ist in Clientumgebungen besonders hilfreich, in denen Code aus dem Internet gedownloadet wird. In dieser Situation ist es unwahrscheinlich, dass dem Code vollständig vertraut wird.

Normalerweise wird Anwendungen vollständig vertraut, die Serviced Components verwenden, daher wird die Codezugriffssicherheit nur eingeschränkt angewendet. Die Enterprise Services erfordern es jedoch, dass der aufrufende Code über die notwendige Berechtigung zum Aufrufen von nicht verwaltetem Code verfügt. Dies bedeutet Folgendes:

- Die Berechtigung für nicht verwalteten Code ist erforderlich, um kontextübergreifende Aufrufe für Serviced Components durchzuführen.
- Wenn es sich bei dem Client einer Serviced Component um eine ASP.NET-Webanwendung handelt, muss diese Anwendung über die Berechtigung für nicht verwalteten Code verfügen.
- Wenn ein Verweis auf eine Serviced Component an nicht vertrauenswürdigen Code übergeben wird, können Methoden nicht vom nicht vertrauenswürdigen Code aufgerufen werden, die für die Serviced Component definiert wurden.

# Konfigurieren der Sicherheit

In diesem Abschnitt wird beschrieben, wie die Sicherheit für Folgendes konfiguriert wird:

- Eine Serviced Component, die in einer Enterprise Services-Serveranwendung (prozessextern) ausgeführt wird.
- Einen ASP.NET-Webanwendungsclient.

## Konfigurieren einer Serveranwendung

Die zum Konfigurieren einer Enterprise Services-Serveranwendung erforderlichen Schritte werden in Abbildung 9.3 aufgeführt.

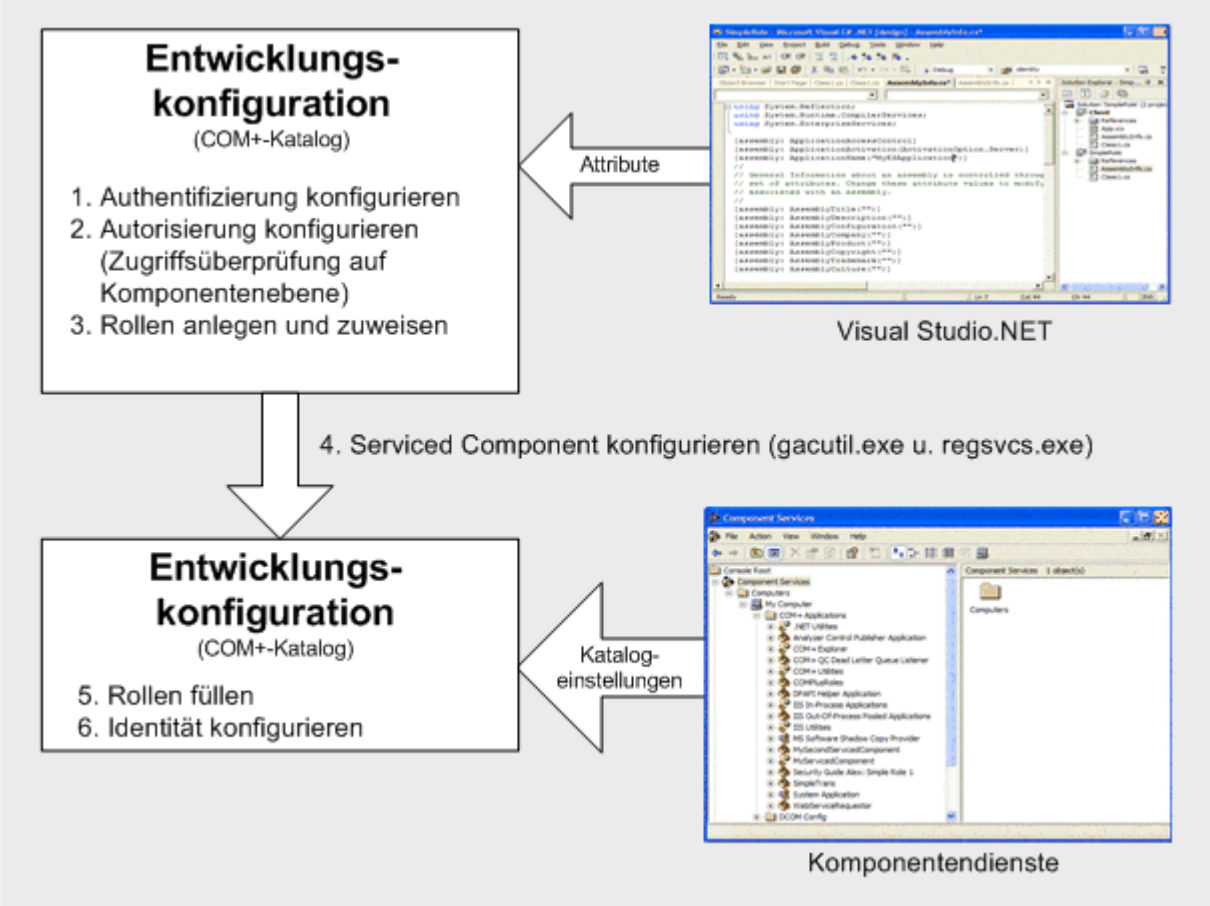


Abbildung 9.3
*Konfigurieren der Enterprise Services-Sicherheit*

## Entwicklungszeit oder Bereitstellungszeitkonfiguration

Sie können die meisten Sicherheitseinstellungen zur Entwicklungszeit im COM+-Katalog konfigurieren, indem Sie die .NET-Attribute in der Assembly verwenden, die die Serviced Component enthält. Diese Attribute werden verwendet, um den COM+-Katalog aufzufüllen, wenn die Serviced Component mit COM+ unter Verwendung des Tools **Regsvcs.exe** registriert wird.

Andere Konfigurationsschritte wie das Auffüllen von Rollen mit Gruppen- und Benutzerkonten von Windows sowie das Konfigurieren als "Ausführen-als"-Identität für die Serveranwendung (Instanz von **Dllhost.exe**) müssen mit dem Verwaltungsprogramm für Komponentendienste (oder programmgesteuert mithilfe eines Skripts) zur Bereitstellungszeit konfiguriert werden.

## Konfigurieren der Authentifizierung

Verwenden Sie das **ApplicationAccessControl**-Assemblyebenenattribut wie unten gezeigt, um die Authentifizierungsebene der Anwendung deklarativ festzulegen.

```
[assembly: ApplicationAccessControl(
              Authentication = AuthenticationOption.Call)]
```

Dies ist mit dem Einstellen des Wertes **Authentifizierungsebene für Aufrufe** auf der Seite **Sicherheit** des Dialogfelds **Eigenschaften** der Anwendung innerhalb der Komponentendienste vergleichbar.

---

**Hinweis:** Die Authentifizierungsebene des Clients wirkt sich auch auf die von der Enterprise Services-Anwendung verwendete Authentifizierungsebene aus, da eine komplexe Aushandlung erfolgt, die immer dazu führt, dass von zwei Einstellungen die höherwertige verwendet wird.

Weitere Informationen zum Konfigurieren der von der ASP.NET-Clientanwendung verwendeten DCOM-Authentifizierungsebene finden Sie weiter unten in diesem Kapitel unter "Konfigurieren einer ASP.NET-Clientanwendung".

Weitere Informationen zu DCOM-Authentifizierungsebenen und Authentifizierungsebenenverhandlungen finden Sie im Abschnitt "Sicherheitskonzepte" in diesem Kapitel.

---

## Konfigurieren der Autorisierung (Zugriffsüberprüfungen auf Komponentenebene)

Sie müssen die folgenden Schritte durchführen, um die feinstufige Autorisierung auf Komponenten-, Schnittstellen- oder Methodenebene zu aktivieren:

- Aktivieren von Zugriffsüberprüfungen auf Anwendungsebene.
  Verwenden Sie das folgende .NET-Attribut, um anwendungsweite Zugriffsüberprüfungen zu aktivieren.

  ```
  [assembly: ApplicationAccessControl(true)]
  ```

  Dies ist mit dem Aktivieren des Kontrollkästchens **Zugriffsüberprüfung für diese Anwendung erzwingen** auf der Seite **Sicherheit** des Dialogfelds **Eigenschaften** der Anwendung innerhalb der Komponentendienste vergleichbar.

  ---

  **Wichtig:** Wenn das Festlegen dieses Attributs fehlschlägt, werden keine Zugriffsüberprüfungen durchgeführt.

  ---

- Konfigurieren der Sicherheitsstufe der Anwendung auf Prozess- und Komponentenebene.
  Für eine sinnvolle, rollenbasierte Sicherheit aktivieren Sie die Zugriffsüberprüfung auf den Prozess- und Komponentenebenen, indem Sie die folgenden .NET-Attribute verwenden.

```
[assembly: ApplicationAccessControl(AccessChecksLevel=
                AccessChecksLevelOption. ApplicationComponent)]
```

  Dies ist mit dem Aktivieren des Kontrollkästchens **Zugriffsüberprüfung auf Prozess- und Komponentenebene durchführen** auf der Seite **Sicherheit** des Dialogfelds **Eigenschaften** der Anwendung innerhalb der Komponentendienste vergleichbar.

---

**Hinweis:** Aktivieren Sie die Zugriffsüberprüfung für Bibliotheksanwendungen immer auf Prozess- und Komponentenebene.

---

- Aktivieren der Zugriffsüberprüfungen auf Komponentenebene.
  Verwenden Sie das **ComponentAccessControl**-Attribut der Klassenebene wie unten gezeigt, um die Zugriffsüberprüfungen auf Komponentenebene zu aktivieren.

```
[ComponentAccessControl(true)]
public class MyServicedComponent : ServicedComponent
{
}
```

  Dies ist mit dem Aktivieren des Kontrollkästchens **Zugriffsüberprüfung auf Komponentenebene erzwingen** auf der Seite **Sicherheit** des Dialogfelds **Eigenschaften** der Komponente innerhalb der Komponentendienste vergleichbar.

---

**Hinweis:** Diese Einstellung zeigt nur Auswirkungen, wenn Sie, wie zuvor beschrieben, die Zugriffsüberprüfung auf Anwendungsebene aktiviert und Zugriffsüberprüfungen auf Prozess- und Komponentenebene konfiguriert haben.

---

## Erstellen und Zuordnen von Rollen

Rollen können auf Anwendungs-, Komponenten- (Klassen-), Schnittstellen- und Methodenebene erstellt und zugeordnet werden.

### Hinzufügen von Rollen zu Anwendungen

Verwenden Sie das **SecurityRole**-Assemblyebenenattribut wie unten gezeigt, um Rollen zu einer Anwendung hinzuzufügen.

```
[assembly:SecurityRole("Employee")]
[assembly:SecurityRole("Manager")]
```

Dies ist mit dem Hinzufügen von Rollen zu einer Anwendung mithilfe des Tools für Komponentendienste vergleichbar.

**Hinweis:** Das Verwenden des **SecurityRole**-Attributs der Assemblyebene ist mit dem Hinzufügen von Rollen zur Anwendung vergleichbar, jedoch ohne sie einzelnen Komponenten, Schnittstellen oder Methoden zuzuordnen. Dies führt dazu, dass die Mitglieder dieser Rollen den Aufbau des Sicherheitsdeskriptors bestimmen, die der Anwendung zugeordnet ist. Dies wird ausschließlich dazu verwendet, um zu ermitteln, wer auf die Anwendung zugreifen (und sie starten) darf.

Wenn Sie eine effektivere rollenbasierte Autorisierung erreichen möchten, ordnen Sie den Komponenten, Schnittstellen und Methoden immer Rollen zu (wie nachfolgend beschrieben).

### Hinzufügen von Rollen zu Komponenten (Klasse)

Wenden Sie das **SecurityRole**-Attribut wie unten gezeigt vor der Klassendefinition an, um Rollen zu einer Komponente hinzuzufügen.

```
[SecurityRole("Manager")]
public class Transfer : ServicedComponent
{
}
```

### Hinzufügen von Rollen zu Schnittstellen

Wenn Sie Rollen auf Schnittstellenebene zuordnen möchten, müssen Sie eine Schnittstellendefinition erstellen und diese innerhalb der Serviced Component-Klasse implementieren. Sie können dann der Schnittstelle Rollen zuordnen, indem Sie das **SecurityRole**-Attribut verwenden.

**Wichtig:** Sie müssen die Klasse zur Entwicklungszeit ebenfalls mit dem **SecureMethod**-Attribut versehen. Dadurch werden die Enterprise Services darüber informiert, dass möglicherweise Sicherheitsdienste auf Methodenebene verwendet werden. Administratoren müssen zur Bereitstellungszeit auch Benutzer zur vom System definierten Rolle **Marshaler** hinzufügen, die automatisch im COM+-Katalog erstellt wird, wenn eine Klasse mit den Komponentendiensten registriert wird, die durch **SecureMethod** gekennzeichnet ist.

Die Verwendung der Rolle **Marshaler** wird im nächsten Abschnitt ausführlicher beschrieben.

Im folgenden Beispiel wird veranschaulicht, wie die Rolle **Manager** zu einer bestimmten Schnittstelle hinzugefügt wird.

```
[SecurityRole("Manager")]
public interface ISomeInterface
{
  void Method1( string message );
  void Method2( int parm1, int parm2 );
}

[ComponentAccessControl]
[SecureMethod]
public class MyServicedComponent : ServicedComponent, ISomeInterface
{
  public void Method1( string message )
  {
    // Implementation
  }
```

```
  public void Method2( int parm1, int parm2 )
  {
    // Implementation
  }
}
```

### Hinzufügen von Rollen zu Methoden

Um sicherzustellen, dass die öffentlichen Methoden einer Klasse im COM+-Katalog erscheinen, müssen Sie explizit eine Schnittstelle implementieren, die die Methoden definiert. Dann müssen Sie das **SecureMethod**-Attribut auf Klassenebene bzw. das Attribut **SecureMethod** oder **SecurityRole** auf Methodenebene verwenden, um die Methoden zu sichern.

---

**Hinweis:** Die Attribute vom Typ **SecureMethod** und **SecurityRole** müssen vor der Methodenimplementierung und nicht innerhalb der Schnittstellendefinition erfolgen.

---

Führen Sie den folgenden Schritt durch, um die Sicherheit auf Methodenebene zu aktivieren:

a. Definieren Sie eine Schnittstelle, die die zu sichernden Methoden enthält. Beispiel:

```
public interface ISomeInterface
{
  void Method1( string message );
  void Method2( int parm1, int parm2 );
}
```

b. Implementieren Sie die Schnittstelle für die Serviced Component-Klasse:

```
[ComponentAccessControl]
public class MyServicedComponent : ServicedComponent, ISomeInterface
{
  public void Method1( string message )
  {
    // Implementation
  }
  public void Method2( int parm1, int parm2 )
  {
    // Implementation
  }
}
```

c. Wenn Sie Rollen administrativ mithilfe des Tools für Komponentendienste konfigurieren möchten, müssen Sie die Klasse wie unten gezeigt mit dem **SecureMethod**-Attribut versehen.

```
[ComponentAccessControl]
[SecureMethod]
public class MyServicedComponent : ServicedComponent, ISomeInterface
{
}
```

d. Alternativ wenden Sie das **SecurityRole**-Attribut auf der Methodenebene an, wenn Sie Rollen zur Entwicklungszeit zu Methoden hinzufügen möchten. In diesem Fall müssen Sie das **SecureMethod**-Attribut nicht auf Klassenebene anwenden (obwohl das **ComponentAccessControl**-Attribut weiterhin vorhanden sein muss, um Zugriffsüberprüfungen auf Komponentenebene konfigurieren zu können).
Im folgenden Beispiel können nur Mitglieder der Rolle **Manager** die Methode **Method1** aufrufen, während Mitglieder der Rollen **Manager** und **Employee** die Methode **Method2** aufrufen können.

```
[ComponentAccessControl]
public class MyServicedComponent : ServicedComponent, ISomeInterface
{
  [SecurityRole("Manager")]
  public void Method1( string message )
  {
    // Implementation
  }
  [SecurityRole("Manager")]
  [SecurityRole("Employee")]
  public void Method2( int parm1, int parm2 )
  {
    // Implementation
  }
}
```

e. Administratoren müssen zur Bereitstellungszeit alle Benutzer zur vordefinierten Rolle **Marshaler** hinzufügen, die den Zugriff auf Methoden oder Schnittstellen der Klasse erfordern.

---

**Hinweis:** Die Enterprise Services-Infrastruktur verwendet eine Reihe von Schnittstellen auf Systemebene, die von allen Serviced Components offen gelegt werden. Diese umfassen **IManagedObject**, **IDisposable** und **IServiceComponentInfo**. Wenn Zugriffsüberprüfungen auf Schnittstellen- oder Methodenebenen aktiviert sind, wird der Enterprise Services-Infrastruktur der Zugriff auf diese Schnittstellen verweigert.

Daher erstellen die Enterprise Services eine spezielle Rolle mit der Bezeichnung **Marshaler** und ordnen die Rolle diesen Schnittstellen zu. Sie können diese Rolle (und die bereits erwähnten Schnittstellen) mit dem Tool für Komponentendienste anzeigen.

Zur Bereitstellungszeit müssen die Anwendungsadministratoren alle Benutzer zur Rolle **Marshaler** hinzufügen, die auf die Methoden oder Schnittstellen der Klasse zugreifen müssen. Sie können diesen Vorgang auf zwei verschiedene Arten automatisieren:

- Schreiben Sie ein Skript, das das Komponentendienste-Objektmodell verwendet, um alle Benutzer anderer Rollen in die Rolle **Marshaler** zu kopieren.
- Schreiben Sie ein Skript, das alle anderen Rollen diesen drei speziellen Schnittstellen zuordnet, und löschen Sie dann die Rolle **Marshaler**.

---

## Registrieren von Serviced Components

Registrieren Sie Serviced Components an folgenden Stellen:

- **Globaler Assemblycache** – Serviced Components, die in COM+-Serveranwendungen verwaltet werden, erfordern die Installation im globalen Assemblycache, was für Bibliotheksanwendungen nicht gilt.

  Führen Sie das Befehlszeilenprogramm **Gacutil.exe** aus, um eine Serviced Component im globalen Assemblycache zu registrieren. Führen Sie den folgenden Befehl aus, um eine Assembly mit der Bezeichnung **MyServicedComponent.dll** im globalen Assemblycache zu registrieren:

  ```
  gacutil –i MyServicedComponent.dll
  ```

  ---

  **Hinweis:** Sie können auch das Konfigurationsprogramm des Microsoft .NET-Frameworks aus der Programmgruppe **Verwaltung** verwenden, um den Inhalt des globalen Assemblycaches anzuzeigen oder zu ändern.

  ---

- **COM+-Katalog** – Führen Sie den folgenden Befehl aus, um eine Assembly mit der Bezeichnung **MyServicedComponent.dll** im COM+-Katalog zu registrieren:

  ```
  regsvcs.exe MyServicedComponent.dll
  ```

  Dieser Befehl führt zur Erstellung einer COM+-Anwendung. Die in der Assembly vorhandenen .NET-Attribute werden dazu verwendet, den COM+-Katalog aufzufüllen.

## Auffüllen von Rollen

Sie können Rollen mithilfe des Tools für Komponentendienste oder unter Verwendung von Skripts auffüllen, mit denen der COM+-Katalog für die Verwendung der COM+-Verwaltungsobjekte programmiert wird.

### Verwenden der Windows-Gruppen

Fügen Sie zu Enterprise Services-Rollen Gruppenkonten von Windows 2000 hinzu, um die maximale Flexibilität zu erreichen. Durch die Verwendung der Windows-Gruppen können Sie ein einzelnes Verwaltungsprogramm (das Verwaltungsprogramm für Benutzer und Computer) effektiv einsetzen, um sowohl die Windows- als auch die Enterprise Services-Sicherheit zu verwalten.

- Erstellen Sie für jede Rolle in der Enterprise Services-Anwendung eine Windows-Gruppe.
- Ordnen Sie jede Gruppe ihrer entsprechenden Rolle zu.

  Wenn Sie z. B. eine Rolle mit der Bezeichnung **Manager** besitzen, erstellen Sie die Windows-Gruppe **Managers**. Weisen Sie die Gruppe **Managers** der Rolle **Manager** zu.
- Nachdem Sie die Gruppen den Rollen zugeordnet haben, verwenden Sie das Verwaltungsprogramm für Benutzer und Computer, um den Gruppen Benutzer hinzuzufügen oder aus diesen zu entfernen.

  Wenn Sie z. B. das Windows 2000-Benutzerkonto **David** der Windows 2000-Gruppe **Managers** hinzufügen, wird **David** der Rolle **Manager** zugeordnet.

▶ **So ordnen Sie Windows-Gruppen zu Enterprise Services-Rollen mithilfe von Komponentendiensten zu:**

1. Erweitern Sie mit dem Tool für Komponentendienste die Anwendung, die die Rollen enthält, die Sie zu Windows 2000-Gruppen hinzufügen möchten.
2. Erweitern Sie den Ordner **Rollen** sowie die bestimmte Rolle, der Sie Windows-Gruppen zuordnen möchten.
3. Wählen Sie den Ordner **Benutzer** unter der bestimmten Rolle aus.
4. Klicken Sie mit der rechten Maustaste auf den Ordner, wählen Sie den Eintrag **Neu** aus, und klicken Sie dann auf **Benutzer**.
5. Fügen Sie im Dialogfeld **Benutzer oder Gruppen auswählen** Gruppen (oder Benutzer) zur Rolle hinzu.

### Weitere Informationen

Weitere Informationen zum Programmieren des COM+-Katalogs unter Verwendung der COM+-Verwaltungsobjekte finden Sie im Abschnitt "Component Development" in MSDN Library unter "Automating COM+ Administration" (englischsprachig).

## Konfigurieren der Identität

Verwenden Sie das Tool für Komponentendienste (oder ein Skript), um die Identität der Enterprise Services-Anwendung zu konfigurieren. Die Eigenschaft für die Identität bestimmt das zum Ausführen der Instanz von **Dllhost.exe** verwendete Konto, das die Anwendung verwaltet.

▶ **So konfigurieren Sie die Identität:**

1. Verwenden Sie das Tool für Komponentendienste zum Auswählen der relevanten Anwendung.
2. Klicken Sie mit der rechten Maustaste auf den Namen der Anwendung, und klicken Sie dann auf **Eigenschaften**.
3. Klicken Sie auf die Registerkarte **Identität**.
4. Klicken Sie auf **Dieser Benutzer**, und geben Sie das zum Ausführen der Anwendung verwendete konfigurierte Dienstkonto an.

### Weitere Informationen

Weitere Informationen zum Auswählen einer geeigneten Identität zum Ausführen einer Enterprise Services-Anwendung finden Sie unter "Auswählen einer Prozessidentität" weiter unten in diesem Kapitel.

# Konfigurieren einer ASP.NET-Clientanwendung

Sie müssen die von Clientanwendungen verwendete DCOM-Authentifizierungsebene und die Identitätswechselebenen konfigurieren, wenn Sie unter Verwendung von DCOM mit Serviced Components kommunizieren.

## Konfigurieren der Authentifizierung

Bearbeiten Sie in der Datei **Machine.config** das **comAuthenticationLevel**-Attribut des Elements <**processModel**>, um die von einer ASP.NET-Webanwendung verwendete Standardauthentifizierungsebene zu konfigurieren.

Die Datei **Machine.config** befindet sich in folgendem Ordner:

```
%windir%\Microsoft.NET\Framework\v1.0.3705\CONFIG
```

Legen Sie für das **comAuthenticationLevel**-Attribut einen der folgenden Werte fest:

```
comAuthenticationLevel=
            "[Default|None|Connect|Call|Pkt|PktIntegrity|PktPrivacy]"
```

### Weitere Informationen

Weitere Informationen zu DCOM-Authentifizierungsebenen finden Sie unter "Authentifizierung" im Abschnitt "Sicherheitskonzepte" weiter unten in diesem Kapitel.

## Konfigurieren des Identitätswechsels

Die vom Client festgelegte Identitätswechselebene bestimmt die Möglichkeiten des Servers hinsichtlich der Identitätswechselebene. Bearbeiten Sie in der Datei **Machine.config** das **comImpersonationLevel**-Attribut des Elements <**processModel**>, um die von einer webbasierten Anwendung beim Kommunizieren mit einer Serviced Component verwendete Standard-Identitätswechselebene zu konfigurieren.

```
comImpersonationLevel="[Default|Anonymous|Identify|Impersonate|Delegate]"
```

### Weitere Informationen

Weitere Informationen zu DCOM-Identitätswechselebenen finden Sie unter "Identitätswechsel" im Abschnitt "Sicherheitskonzepte" weiter unten in diesem Kapitel.

# Konfigurieren der Identitätswechselebenen für eine Enterprise Services-Anwendung

Wenn eine Serviced Component in einer Anwendung eine Serviced Component in einer zweiten (Server-) Anwendung aufrufen muss, müssen Sie die Identitätswechselebene für die Clientanwendung konfigurieren.

---

**Wichtig:** Die für eine Enterprise Services-Anwendung konfigurierte Identitätswechselebene (auf der Seite **Sicherheit** des Dialogfelds **Eigenschaften**) ist die von ausgehenden Aufrufen verwendete Identitätswechselebene, die von Komponenten in der Anwendung durchgeführt werden. Sie wirkt sich nicht darauf aus, ob Serviced Components in der Anwendung den Identitätswechsel durchführen. Wenn Sie für Clients innerhalb einer Serviced Component einen Identitätswechsel durchführen möchten, müssen Sie programmatische Identitätswechselverfahren verwenden, wie unter "Übermitteln des ursprünglichen Aufrufers" weiter unten in diesem Kapitel beschrieben.

---

Verwenden Sie das **ApplicationAccessControl**-Assemblyebenenattribut wie unten gezeigt, um die Identitätswechselebene der Anwendung deklarativ festzulegen.

```
[assembly: ApplicationAccessControl(
                ImpersonationLevel=ImpersonationLevelOption.Identify)]
```

Dies ist mit dem Einstellen des Wertes **Identitätswechselebene** auf der Seite **Sicherheit** des Dialogfelds **Eigenschaften** der Anwendung innerhalb der Komponentendienste vergleichbar.

# Programmieren der Sicherheit

Die Sicherheitsfunktionen der Enterprise Services stehen für .NET-Komponenten zur Verfügung, die die Klassen **ContextUtil**, **SecurityCallContext** und **SecurityIdentity** verwenden.

## Programmatische rollenbasierte Sicherheit

Sie können die Rollenmitgliedschaft programmatisch mit der Methode **IsCallerInRole** der Klasse **ContextUtil** testen, um feinstufige Autorisierungsentscheidungen zu treffen. Bevor Sie diese Methode aufrufen, müssen Sie immer sicherstellen, dass Zugriffsüberprüfungen auf Komponentenebene aktiviert sind, wie im nachfolgenden Codefragment gezeigt wird. Wenn die Sicherheit deaktiviert ist, gibt **IsCallerInRole** immer **True** (Wahr) zurück.

```
public void Transfer(string fromAccount, string toAccount, double amount)
{
  // Check that security is enabled
  if (ContextUtil.IsSecurityEnabled)
  {
    // Only Managers are allowed to transfer sums of money in excess of $1000
    if (amount > 1000)
    {
      if (ContextUtil.IsCallerInRole("Manager"))
      {
        // Caller is authorized
      }
      else
      {
        // Caller is unauthorized
      }
    }
}
```

## Identifizieren von Aufrufern

Im folgenden Beispiel wird gezeigt, wie alle Upstreamaufrufer innerhalb einer Serviced Component identifiziert werden.

```
[ComponentAccessControl]
public class MyServicedComponent : ServicedComponent
{
  public void ShowCallers()
  {
    SecurityCallContext context = SecurityCallContext.CurrentCall;
    SecurityCallers callers = context.Callers;
    foreach(SecurityIdentity id in callers)
    {
      Console.WriteLine(id.AccountName);
    }
  }
}
```

**Hinweis:** Die Identität des ursprünglichen Aufrufers steht über die Eigenschaft **SecurityCallContext.OriginalCaller** zur Verfügung.

# Auswählen einer Prozessidentität

Für Server aktivierte Enterprise Services-Anwendungen werden in einer Instanz des Prozesses **Dllhost.exe** ausgeführt. Sie müssen das zum Ausführen des Prozesses verwendete Konto im COM+-Katalog mithilfe des Tools für Komponentendienste konfigurieren.

**Hinweis:** Sie können die "Ausführen-als"-Identität nicht festlegen, indem Sie ein .NET-Attribut verwenden.

## Niemals als interaktiver Benutzer ausführen

Führen Sie Serveranwendungen nicht unter Verwendung der Identität des interaktiv angemeldeten Benutzers aus (dies ist die Standardeinstellung). Es gibt zwei Hauptgründe, um dies zu vermeiden:

- Die (Zugriffs-)Rechte der Anwendung variieren und hängen davon ab, wer momentan interaktiv auf dem Server angemeldet ist. Wenn ein Administrator angemeldet ist, verfügt die Anwendung über Administratorrechte.
- Wenn die Anwendung gestartet wird, während ein Benutzer interaktiv angemeldet ist und dieser sich dann abmeldet, wird die Serveranwendung heruntergefahren. Es kann nicht erneut gestartet werden, bis sich ein anderer Benutzer interaktiv anmeldet.

Die interaktive Benutzereinstellung ist für Entwickler für den Einsatz zur Entwicklungszeit gedacht und sollte nicht als Einstellung für die Bereitstellung betrachtet werden.

## Verwenden eines benutzerdefinierten Kontos mit minimalen Rechten

Erstellen Sie ein Konto mit minimalen Rechten, um die Bedrohung durch das Angreifen von Prozessen zu verringern. Wenn es einem entschlossener Angreifer gelingt, den Serverprozess anzugreifen, kann dieser die (Zugriffs-)Rechte problemlos übernehmen, die dem Prozesskonto gewährt wurden. Ein Konto, das mit den minimalen Rechten konfiguriert wurde, schränkt den potenziell anzurichtenden Schaden ein.

Wenn Sie mit dem Prozesskonto auf Netzwerkressourcen zugreifen müssen, muss der Remotecomputer das Prozesskonto authentifizieren können. In diesem Szenario bieten sich Ihnen zwei Optionen:

- Sie können ein Domänenkonto verwenden, wenn sich die beiden Computer in derselben oder in vertrauten Domänen befinden.
- Sie können ein lokales Konto verwenden und dann ein dupliziertes Konto (mit demselben Benutzernamen und demselben Kennwort) auf dem Remotecomputer erstellen. Bei dieser Option müssen Sie sicherstellen, dass die Kennwörter der beiden Konten synchron bleiben.

  Es kann erforderlich sein, den Ansatz mit einem duplizierten lokalen Konto zu verwenden, wenn sich der Remotecomputer in einer separaten Domäne (ohne Vertrauensstellung) oder hinter einem Firewall (bei dem geschlossene Ports die Windows-Authentifizierung nicht gewähren) befindet.

# Zugreifen auf Netzwerkressourcen

Ihre Serviced Components müssen möglicherweise auf Netzwerkressourcen zugreifen. Dabei ist es wichtig, dass Sie in der Lage sind, Folgendes zu kennzeichnen:

- Die Ressourcen, auf die die Komponenten zugreifen müssen. Beispielsweise Dateien auf Dateifreigaben, Datenbanken, andere DCOM-Server, Active Directory®-Verzeichnisdienstobjekte usw.
- Die zum Durchführen des Ressourcenzugriffs verwendete Identität. Wenn Ihre Serviced Component auf Remoteressourcen zugreift, muss die verwendete Identität (bei der es sich standardmäßig um die Prozessidentität handelt) vom Remotecomputer authentifiziert werden können.

---

**Hinweis:** Weitere Informationen zum Zugreifen auf SQL Server-Remotedatenbanken finden Sie in Kapitel 12, "Datenzugriffssicherheit".

---

Sie können über eine Komponente innerhalb einer Enterprise Services-Anwendung auf Remoteressourcen zugreifen, indem Sie eine der folgenden Identitäten verwenden:

- Der ursprüngliche Aufrufer (wenn Sie den Identitätswechsel explizit unter Verwendung von **CoImpersonateClient** durchführen)
- Die aktuelle Prozessidentität (für Serveranwendungen im COM+-Katalog konfiguriert)
- Ein bestimmtes Dienstkonto

## Verwenden des ursprünglichen Aufrufers

Sie müssen Folgendes durchführen, um die Identität des ursprünglichen Aufrufers für den Zugriff auf Remoteressourcen zu verwenden:

- Wechseln Sie die Identität des ursprünglichen Aufrufers programmatisch durch Aufrufen von **CoImpersonateClient**.
- Sie müssen in der Lage sein, den Sicherheitskontext des Aufrufers vom Anwendungsserver, der die Enterprise Services-Anwendung verwaltet, zum Remotecomputer zu delegieren. Hierbei wird davon ausgegangen, dass Sie zwischen der Enterprise Services-Anwendung und der Clientanwendung die Kerberos-Authentifizierung verwenden.

---

**Skalierbarkeitswarnung:** Wenn Sie auf die Datendienstebene der Anwendung mithilfe der gewechselten Identität des ursprünglichen Aufrufers zugreifen, beeinflussen Sie die Skalierbarkeit der Anwendung erheblich, da das Pooling der Datenbankverbindung ineffektiv erfolgt. Die Ineffektivität rührt daher, dass dann der Sicherheitskontext jeder Datenbankverbindung mit vielen einzelnen Aufrufern verbunden ist.

---

### Weitere Informationen

Weitere Informationen zum Identitätswechsel von Aufrufern finden Sie unter "Übermitteln des ursprünglichen Aufrufers" weiter unten in diesem Kapitel.

## Verwenden der aktuellen Prozessidentität

Wenn Ihre Anwendung zum Ausführen als Serveranwendung konfiguriert ist, können Sie die konfigurierte Prozessidentität für den Zugriff auf Remoteressourcen verwenden (dies ist der Standardfall).

Wenn Sie das Serverprozesskonto für den Zugriff auf Remoteressourcen verwenden möchten, müssen Sie eine der folgenden Möglichkeiten wählen:

- Ausführen der Serveranwendung mithilfe eines Domänenkontos mit minimalen Rechten. Dies setzt voraus, dass sich die Client- und Servercomputer in derselben oder in vertrauten Domänen befinden.
- Duplizieren Sie das Prozesskonto, indem Sie denselben Benutzernamen und dasselbe Kennwort auf dem Remotecomputer verwenden.

Wenn die einfache Verwaltung Ihr Hauptanliegen darstellt, sollten Sie ein Domänenkonto mit minimalen Rechten verwenden.

Wenn Ihre Anwendung zum Ausführen als Bibliotheksanwendung konfiguriert ist, wird die Prozessidentität vom Hostprozess geerbt (bei dem es sich häufig um eine webbasierte Anwendung handelt). Weitere Informationen zum Verwenden der ASP.NET-Prozessidentität für den Zugriff auf Remoteressourcen finden Sie in Kapitel 8, "ASP.NET-Sicherheit".

## Verwenden eines bestimmten Dienstkontos

Ihre Enterprise Services-Anwendung kann mithilfe eines speziell konfigurierten Dienstkontos auf Remoteressourcen zugreifen (d. h. ein Nicht-Benutzerkonto von Windows). Dieser Ansatz wird unter Windows 2000 nicht empfohlen, da er davon abhängt, dass Sie die API **LogonUser** aufrufen.

Die Verwendung von **LogonUser** unter Windows 2000 zwingt Sie, dem Enterprise Services-Prozesskonto das Recht **Als Teil des Betriebssystems handeln** zu gewähren. Dadurch wird die Sicherheit Ihrer Anwendung erheblich verringert.

---

**Hinweis:** Microsoft Windows Server 2003 wird diese Einschränkung aufheben.

---

# Übermitteln des ursprünglichen Aufrufers

Ausgehende Aufrufe von Serviced Components (z. B. für den Zugriff auf lokale oder Remoteressourcen) werden standardmäßig mit dem vom Hostprozess erhaltenen Sicherheitskontext durchgeführt. Für Serveranwendungen ist dies die "Ausführen-als"-Identität. Bei Bibliotheksanwendungen ist dies die Identität des (Host-)Clientprozesses (z. B. **Aspnet_wp.exe**, wenn der Client eine ASP.NET-Webanwendung ist).

▶ **So übermitteln Sie den Kontext des ursprünglichen Aufrufers durch eine Enterprise Services-Anwendung:**

1. Rufen Sie **CoImpersonateClient** auf.

   Dadurch wird ein Identitätswechseltoken für Threads erstellt und dieses dem aktuellen Thread zugeordnet.
2. Führen Sie die Operation durch (Zugriff auf lokale oder Remoteressourcen).

   Wenn der Identitätswechsel aktiviert ist, wird der ausgehende Aufruf unter Verwendung des Sicherheitskontextes des Clients durchgeführt (wie durch das Identitätswechseltoken definiert).

   Beim Zugriff auf lokale Ressourcen muss der Aufrufer (Clientprozess) zumindest den Identitätswechsel auf der Ebene **Impersonate** festgelegt haben. Beim Zugriff auf Remoteressourcen muss der Aufrufer zumindest den Identitätswechsel auf der Ebene **Delegate** festgelegt haben.

   Wenn es sich bei dem Aufrufer um eine ASP.NET-Webanwendung handelt, wird als Standard für die Identitätswechselebene von ASP.NET-Workerprozessen die Einstellung **Impersonate** verwendet. Daher müssen Sie diese Standardeinstellung in **Delegate** ändern (auf dem Clientcomputer im Element <**processModel**> der Datei **Machine.config**), um den ursprünglichen Aufrufer an einen nachgeschalteten Remotecomputer zu übermitteln.

**Hinweis:** Sie müssen die Kerberos-Authentifizierung für Konten verwenden, die für die Delegierung konfiguriert sind, um den Sicherheitskontext des ursprünglichen Aufrufers zum Zugreifen auf Remoteressourcen zu verwenden. Das zum Ausführen der Enterprise Services-Anwendung verwendete Konto muss außerdem in Active Directory als **Konto wird für Delegierungszwecke vertraut** gekennzeichnet sein.

3. Beenden Sie den Identitätswechsel durch Aufrufen von **CoRevertToSelf**.
   Dadurch wird das Identitätswechseltoken entfernt. Alle anschließenden Aufrufe der aktuellen Methode verwenden den Prozesssicherheitskontext. Wenn Sie **CoRevertToSelf** nicht aufrufen, wird es beim Beenden der Methode implizit vom Laufzeitmodul aufgerufen.

**Hinweis:** Die Identität des ursprünglichen Aufrufers wird automatisch an eine Enterprise Services-Anwendung übermittelt und steht bei Verwendung von **SecurityCallContext.OriginalCaller** zur Verfügung. Dies kann besonders für Überwachungsaktivitäten hilfreich sein.

## Aufrufen von "CoImpersonateClient"

**CoImpersonateClient** (und **CoRevertToSelf**) befinden sich in der Datei **OLE32.dll**. Sie müssen deren Definitionen unter Verwendung des **DllImport**-Attributs importieren, damit Sie sie über **P/Invoke** aufrufen können. Dies wird im nachfolgenden Code veranschaulicht.

```
class COMSec
{
  [DllImport("OLE32.DLL", CharSet=CharSet.Auto)]
  public static extern uint CoImpersonateClient();

  [DllImport("OLE32.DLL", CharSet=CharSet.Auto)]
  public static extern uint CoRevertToSelf();
}
. . .
void SomeMethod()
{
  // To flow the original caller's security context and use it to access local
  // or remote resources, start impersonation
  COMSec.CoImpersonateClient();
  // Perform operations as the caller
  // Code here uses the context of the caller – not the context of the process
  . . .
  COMSec.CoRevertToSelf();
  // Code here reverts to using the process context
}
```

### Weitere Informationen

Weitere Informationen zum Konfigurieren eines vollständigen Kerberos-Delegierungsszenarios, das die Übermittlung des Sicherheitskontextes des ursprünglichen Aufrufers über eine ASP.NET-Webanwendung, eine Enterprise Services-Anwendung und eine Datenbank veranschaulicht, finden Sie unter "Übermitteln des ursprünglichen Aufrufers" in Kapitel 5, "Intranetsicherheit".

# RPC-Verschlüsselung

Verwenden Sie zwischen Client und Server die Authentifizierungsebene für RPC-Paketsicherheit, um die von einer Clientanwendung über DCOM an eine Remote-Serviced Component gesendeten Daten zu sichern. Dadurch wird die Vertraulichkeit und Integrität für Nachrichten bereitgestellt.

Sie müssen die Authentifizierungsebene auf dem Client und dem Server konfigurieren.

Legen Sie für das **comAuthenticationLevel**-Attribut des Elements <**processModel**> in der Datei **Machine.config** den Wert **PktPrivacy** fest.

Legen die Authentifizierungsebene für die Anwendungsebene entweder mithilfe des Tools für Komponentendienste oder über das folgende .NET-Attribut innerhalb der Serviced Component-Assembly fest, um eine Enterprise Services-Serveranwendung zu konfigurieren.

```
[assembly: ApplicationAccessControl(
              Authentication = AuthenticationOption.Privacy)]
```

### Weitere Informationen

- Weitere Informationen zum Konfigurieren der Sicherheit (einschließlich der Authentifizierungsebenen) finden Sie unter "Konfigurieren der Sicherheit" weiter oben in diesem Kapitel.
- Weitere Informationen zu RPC/DCOM-Authentifizierungsebenen finden Sie unter "Authentifizierung" weiter unten in diesem Kapitel.
- Weitere Informationen zur Verhandlung auf der Authentifizierungsebene finden Sie unter "Verhandeln auf der Authentifizierungsstufe" weiter unten in diesem Kapitel.

# Erstellen von Serviced Components

Eine schrittweise Anleitung zum Erstellen einer Serviced Component finden Sie unter "Vorgehensweise: Verwenden von rollenbasierter Sicherheit mit Enterprise Services" im Abschnitt "Referenz" dieses Handbuchs.

### Probleme beim Sperren von DLLs

Führen Sie die folgenden Schritte beim erneuten Erstellen einer Serviced Component durch, wenn die DLL gesperrt ist:

- Verwenden Sie die Komponentendienste, um die COM+-Serveranwendung herunterzufahren.
- Wenn Sie eine Bibliotheksanwendung erstellen, ist die Anwendung möglicherweise weiterhin im Prozess **Aspnet_wp.exe** geladen. Führen Sie **IISReset** über eine Eingabeaufforderung aus, oder verwenden Sie den Task-Manager, um den Prozess **Aspnet_wp.exe** anzuhalten.
- Verwenden Sie das Tool **FileMon.exe** von www.sysinternals.com, um Probleme mit gesperrten Dateien zu beheben.

### Versionserstellung

Das von der Entwicklungsumgebung Microsoft Visual Studio® .NET beim Erstellen eines neuen Projekts generierte **AssemblyVersion**-Standardattribut ist nachfolgend dargestellt:

```
[assembly: AssemblyVersion("1.0.*")]
```

Bei jedem erneuten Erstellen des Projekts wird eine neue Assemblyversion generiert. Dies führt auch zum Generieren eines neuen Klassenbezeichners (CLSID), um die Serviced Component-Klassen zu kennzeichnen. Wenn Sie die Assembly wiederholt mit den Komponentendiensten unter Verwendung der Datei **Regsvcs.exe** registriert haben, werden duplizierte Komponenten (genau genommen Klassen) mit unterschiedlichen CLSIDs unter dem Ordner **Komponenten** angezeigt.

Während dies der strikten Semantik der COM-Versionserstellung entspricht und Probleme bei vorhandenen verwalteten sowie nicht verwalteten Clients verhindert, kann es sich während der Entwicklung als störend erweisen.

Während der Tests und der Entwicklung sollten Sie erwägen, eine explizite Version unter Verwendung des **AssemblyVersion**-Attributs der Assemblyebene festzulegen, wie nachfolgend gezeigt:

```
[assembly: AssemblyVersion("1.0.0.1")]
```

Diese Einstellung verhindert das Generieren einer neuen CLSID bei jeder nachfolgenden Projekterstellung. Sie können auch das Problem für die Schnittstellenbezeichner (Interface Identifiers, IIDs) beheben. Wenn Ihre Klasse explizite Schnittstellen implementiert, können Sie den Schnittstellenbezeichner für eine gegebene Schnittstelle, wie nachfolgend gezeigt, mithilfe des GUID-Attributs ändern.

```
[Guid("E1FBF27E-9F11-474d-8DF6-58916F798E9D")]
public interface IMyInterface
{
}
```

▶ **So generieren Sie neue GUIDs:**

1. Klicken Sie im Menü **Extras** von Visual Studio .NET auf **GUID erstellen**.
2. Klicken Sie auf **Registrierungsformat**.
3. Klicken Sie auf **Neue GUID**.
4. Klicken Sie auf **Kopieren**.
5. Fügen Sie die GUID aus der Zwischenablage in den Quellcode ein.

---

**Wichtig:** Vor dem Bereitstellen Ihrer Serviced Component-Assembly zum Testen und Produzieren entfernen Sie alle geänderten GUIDs und kehren zu einem automatisierten Versionserstellungsmechanismus zurück (z. B. durch Verwenden von "1.0.*"). Wenn Sie diesen Vorgang nicht durchführen, erhöht sich die Wahrscheinlichkeit, dass eine neue Version Ihrer Komponente bei vorhandenen Clients zu Problemen führen kann.

---

### Weitere Informationen

Weitere Informationen zur Versionserstellung für die Bereitstellung finden Sie in MSDN unter Understanding Enterprise Services (COM+) in .NET (englischsprachig).

## "QueryInterface"-Ausnahmen

Wenn ein Aufruf von **QueryInterface** für die Schnittstelle **IRoleSecurity** fehlschlägt, zeigt dies an, dass Sie eine Schnittstellendefinition innerhalb Ihrer Assembly aktualisiert, die Assembly jedoch nicht mit den Komponentendiensten unter Verwendung von **Regsvcs.exe** erneut registriert haben.

---

**Wichtig:** Bei jeder Ausführung von **Regsvcs.exe** müssen Sie die "Ausführen-als"-Identität einer Serveranwendung erneut konfigurieren und die Benutzer wieder zu Gruppen hinzufügen. Sie können ein einfaches Skript erstellen, um diese Aufgabe zu automatisieren.

---

# DCOM und Firewalls

Windows 2000 (SP3 oder QFE 18.1) oder Windows Server 2003 ermöglichen die Konfiguration von Enterprise Services-Anwendungen für die Verwendung eines statischen Endpunktes. Wenn ein Firewall den Client vom Server trennt, müssen Sie in dem Firewall nur zwei Ports öffnen. Insbesondere muss Port 135 für Remoteprozeduraufrufe und ein Port für die Enterprise Services-Anwendung geöffnet werden.

Alternativ können Sie Ihre Enterprise Services-Anwendung als Webdienst offen legen. Dadurch können Sie Serviced Components unter Verwendung von SOAP über Port 80 aktivieren und aufrufen. Das Hauptproblem bei diesem Ansatz ist, dass er es Ihnen nicht ermöglicht, den Transaktionskontext vom Client an den Server zu übermitteln. Sie müssen Ihre Transaktion auf der Remote-Serviced Component starten.

### Weitere Informationen

Weitere Informationen finden Sie in den folgenden Knowledge Base-Artikeln:

- Artikel Q312960, "Cannot Set Fixed Endpoint for a COM+ Application" (US)
- Artikel Q259011, "SAMPLE: A Simple DCOM Client Server Test Application" (US)
- Artikel Q248809, "PRB: DCOM Does Not Work over NAT-Based Firewall" (US)
- Artikel Q250367, "INFO: Configuring Microsoft Distributed Transaction Coordinator (DTC) to Work Through a Firewall" (US)
- Artikel Q154596, "HOWTO: Configure RPC Dynamic Port Allocation to Work w/ Firewall" (US)

# Aufrufen von Serviced Components über ASP.NET

In diesem Abschnitt werden die Hauptthemen bezüglich der Situation beschrieben, in der eine ASP.NET-Anwendung eine Serviced Component aufruft.

## Aufruferidentität

Wenn Sie eine Serviced Component über eine ASP.NET-Anwendung aufrufen, wird die Sicherheitsidentität von der Win32®-Threadidentität der Anwendung abgerufen. Wenn die Webanwendung für den Identitätswechsel des Aufrufers konfiguriert ist, handelt es sich hierbei um die Identität des Aufrufers. Andernfalls ist es die ASP.NET-Prozessidentität (standardmäßig ASPNET).

Bei einer ASP.NET-Anwendung können Sie die aktuelle Win32-Threadidentität durch Aufrufen von **WindowsIdentity.GetCurrent()** abrufen.

Bei einer Serviced Component können Sie die Identität des ursprünglichen Aufrufers durch die Verwendung von **SecurityCallContext.OriginalCaller** abrufen.

## Verwenden der Windows-Authentifizierung und des Identitätswechsels innerhalb der webbasierten Anwendung

Sie müssen die Windows-Authentifizierung verwenden und den Identitätswechsel aktivieren, um eine sinnvolle, rollenbasierte Sicherheit in Ihrer Enterprise Services-Anwendung zu ermöglichen. Dadurch wird sichergestellt, dass die Serviced Components den ursprünglichen Aufrufer authentifizieren und Autorisierungsentscheidungen auf Basis der Identität des ursprünglichen Aufrufers treffen können.

## Konfigurieren der Authentifizierung und des Identitätswechsels in "Machine.config"

Die DCOM-Authentifizierungsebenen werden zwischen dem Client (z. B. einer webbasierten Anwendung) und dem Server (der Enterprise Services-Anwendung) verhandelt. Dabei wird die höhere der beiden Sicherheitseinstellungen verwendet.

Konfigurieren Sie die ASP.NET-Authentifizierungsebenen über das **comAuthentication**-Attribut des Elements <**processModel**> in der Datei **Machine.config**.

Die Identitätswechselebenen werden vom Client gesteuert (z. B. eine webbasierte Anwendung). Der Client kann den Grad für den Identitätswechsel bestimmten, den der Server verwenden darf.

Konfigurieren Sie die ASP.NET-Identitätswechselebenen (für alle ausgehenden DCOM-Aufrufe) über das **comImpersonationLevel**-Attribut des Elements <**processModel**> in der Datei **Machine.config**.

## Konfigurieren von Schnittstellenproxys

Die für einzelne Schnittstellenproxys geltenden Sicherheitseinstellungen werden von den Standardsicherheitseinstellungen für die Prozessebene abgerufen. Bei ASP.NET werden die Standardsicherheitseinstellungen (wie die Identitätswechselebene und die Authentifizierungsebene) in der Datei **Machine.config** konfiguriert, wie bereits zuvor in diesem Kapitel beschrieben.

Bei Bedarf können Sie die von einem einzelnen Schnittstellenproxy verwendeten Sicherheitseinstellungen ändern. Wenn Ihre ASP.NET-Anwendung z. B. mit einer Serviced Component kommuniziert, die zwei Schnittstellen offen legt, wobei aber nur über eine Schnittstelle vertrauliche Daten übertragen werden, entscheiden Sie sich möglicherweise für die Verwendung der Verschlüsselungsunterstützung, die von der Authentifizierungsebene für Paketsicherheit nur für die Schnittstelle mit den vertraulichen Daten bereitgestellt wird, während Sie für die andere Schnittstelle z. B. die Paketauthentifizierung verwenden. Das bedeutet, dass der Leistungseinbruch ausbleibt, der einer Verschlüsselung für beide Schnittstellen anhängt.

Zusammengenommen wird die Gruppe der Sicherheitseinstellungen, die für einen Schnittstellenproxy gelten, als Sicherheitsblanket bezeichnet. COM stellt die folgenden Funktionen bereit, um Ihnen das Abfragen und Ändern von Sicherheitsblanketeinstellungen für einen einzelnen Schnittstellenproxy zu ermöglichen:

- **CoQueryProxyBlanket**
- **CoSetProxyBlanket**
- **CoCopyProxy**

Sie müssen **P/Invoke** zum Aufrufen dieser Funktionen über eine ASP.NET-Webanwendung (der DCOM-Client) verwenden. Der folgende Code veranschaulicht, wie eine bestimmte Schnittstelle für die Verwendung der Authentifizierungsebene für Paketsicherheit (die die Verschlüsselung ermöglicht) konfiguriert wird. Dieser Code kann von einer ASP.NET-Webanwendung verwendet werden, die mit einer Remote-Serviced Component kommuniziert.

```
// Define a wrapper class for the P/Invoke call to CoSetProxyBlanket
class COMSec
{
  // Constants required for the call to CoSetProxyBlanket
  public const uint RPC_C_AUTHN_DEFAULT          = 0xFFFFFFFF;
  public const uint RPC_C_AUTHZ_DEFAULT          = 0xFFFFFFFF;
  public const uint RPC_C_AUTHN_LEVEL_PKT_PRIVACY = 6;
  public const uint RPC_C_IMP_LEVEL_DEFAULT      = 0;
  public const uint COLE_DEFAULT_AUTHINFO        = 0xFFFFFFFF;
  public const uint COLE_DEFAULT_PRINCIPAL       = 0;
  public const uint EOAC_DEFAULT                 = 0x800;

  // HRESULT  CoSetProxyBlanket( IUnknown * pProxy,
  //                             DWORD dwAuthnSvc,
  //                             DWORD dwAuthzSvc,
  //                             WCHAR * pServerPrincName,
  //                             DWORD dwAuthnLevel,
  //                             DWORD dwImpLevel,
  //                             RPC_AUTH_IDENTITY_HANDLE pAuthInfo,
  //                             DWORD dwCapabilities );
[DllImport("OLE32.DLL", CharSet=CharSet.Auto)]
public unsafe static extern uint CoSetProxyBlanket(
                                     IntPtr pProxy,
                                     uint dwAuthnSvc,
                                     uint dwAuthzSvc,
                                     IntPtr pServerPrincName,
                                     uint dwAuthnLevel,
                                     uint dwImpLevel,
                                     IntPtr pAuthInfo,
                                     uint dwCapababilities);
} // end class COMSec


// Code to call CoSetProxyBlanket
void CallComponent()
{
  // This is the interface to configure
  Guid IID_ISecureInterface = new Guid("c720ff19-bec1-352c-bb4b-e2de10b858ba");
  IntPtr pISecureInterface;

  // Instantiate the serviced component
  CreditCardComponent comp = new CreditCardComponent();
  // Get its IUnknown pointer
  IntPtr pIUnk = Marshal.GetIUnknownForObject(comp);
  // Get the interface to configure
  Marshal.QueryInterface(pIUnk, ref IID_ISecureInterface,
                         out pISecureInterface);
  try
  {
    // Configure the interface proxy and set packet privacy authentication
    uint hr = COMSec.CoSetProxyBlanket( pISecureInterface,
                                        COMSec.RPC_C_AUTHN_DEFAULT,
                                        COMSec.RPC_C_AUTHZ_DEFAULT,
```

```
                                        IntPtr.Zero,
                                        COMSec.RPC_C_AUTHN_LEVEL_PKT_PRIVACY,
                                        COMSec.RPC_C_IMP_LEVEL_DEFAULT,
                                        IntPtr.Zero,
                                        COMSec.EOAC_DEFAULT );
    ISecureInterface secure = (ISecureInterface)comp;
    // The following call will be encrypted as ISecureInterface is configured
    // for packet privacy authentication. Other interfaces use the process
    // level defaults (normally packet authentication).
    secure.ValidateCreditCard("123456789");
  }
  catch (Exception ex)
  {
  }
}
```

### Weitere Informationen

- Weitere Informationen zum Konfigurieren einer ASP.NET-Clientanwendung, damit diese Serviced Components aufruft, finden Sie unter "Konfigurieren einer ASP.NET-Clientanwendung" weiter oben in diesem Kapitel.
- Weitere Informationen zu DCOM-Authentifizierungsebenen finden Sie unter "Authentifizierung" weiter unten in diesem Kapitel.
- Weitere Informationen zu DCOM-Identitätswechselebenen finden Sie unter "Identitätswechsel" weiter unten in diesem Kapitel.
- Weitere Informationen zur Verwendung der Windows-Authentifizierung und des Identitätswechsels innerhalb einer webbasierten Anwendung finden Sie in Kapitel 8, "ASP.NET-Sicherheit".

# Sicherheitskonzepte

Dieser Abschnitt bietet eine kurze Übersicht über die Sicherheitskonzepte von Enterprise Services. Viele Konzepte werden Ihnen bekannt sein, wenn Sie bereits Erfahrungen mit COM+ haben.

Hintergrundinformationen zu Enterprise Services finden Sie im MSDN-Artikel "Understanding Enterprise Services (COM+) in .NET" (englischsprachig).

Nachfolgend werden die Sicherheitskonzepte zusammengefasst, die Ihnen vertraut sein sollten.

- Sicherheitseinstellungen für Serviced Components und Enterprise Services-Anwendungen werden im COM+-Katalog verwaltet. Die meisten Einstellungen können mithilfe der .NET-Attribute konfiguriert werden. Alle Einstellungen können unter Verwendung des Verwaltungsprogramms für Komponentendienste oder des Entwicklungssystemskripts von Microsoft Visual Basic® Scripting Edition konfiguriert werden.
- Die Autorisierung wird über Enterprise Services (COM+)-Rollen bereitgestellt, die Gruppen- oder Benutzerkonten von Windows enthalten können. Hierbei handelt es sich nicht um .NET-Rollen.
    - Die rollenbasierte Sicherheit kann auf Anwendungs-, Schnittstellen-, Klassen- und Methodenebene zugeordnet werden.
    - Imperative Rollenüberprüfungen können innerhalb von Methoden unter Verwendung der Methode **IsCallerInRole** der Klasse **ContextUtil** programmatisch durchgeführt werden.

- Die effektive rollenbasierte Autorisierung in einer Enterprise Services-Anwendung hängt von einer Windows-Identität ab, die zum Aufrufen von Serviced Components verwendet wird.
  - Dies kann von Ihnen innerhalb einer ASP.NET-Webanwendung die Verwendung der Windows-Authentifizierung zusammen mit dem Identitätswechsel erfordern – wenn die Webanwendung Serviced Components aufruft, die von Enterprise Services (COM+)-Rollen abhängen.
  - Wenn Sie eine Serviced Component über eine ASP.NET-Webanwendung oder einen Webdienst aufrufen, wird die für den ausgehenden DCOM-Aufruf verwendete Identität durch die Win32-Threadidentität bestimmt, wie von **WindowsIdentity.GetCurrent()** definiert.
- Serviced Components können in Server- oder Bibliotheksanwendungen ausgeführt werden.
  - Serveranwendungen werden in separaten Instanzen der Datei **Dllhost.exe** ausgeführt.
  - Bibliotheksanwendungen werden Prozessadressraum des Clients ausgeführt.
  - Die rollenbasierte Autorisierung funktioniert für Server- und Bibliotheksanwendungen ähnlich, obwohl es hinsichtlich der Sicherheit einige geringfügige Unterschiede zwischen Bibliotheks- und Serveranwendungen gibt. Weitere Informationen finden Sie unter "Sicherheit für Server- und Bibliotheksanwendungen" weiter oben in diesem Kapitel.
- Die Authentifizierung wird von den zugrunde liegenden Diensten von DCOM und RPC bereitgestellt. Mit der Kombination der Authentifizierungsebene des Clients und des Servers wird die sich ergebende Authentifizierungsebene bestimmt, die für die Kommunikation mit der Serviced Component verwendet wird.
- Der Identitätswechsel wird innerhalb der Clientanwendung konfiguriert. Er bestimmt die Möglichkeiten des Servers für den Identitätswechsel.

## Enterprise Services (COM+)-Rollen und .NET-Rollen

Enterprise Services (COM+)-Rollen werden zum Darstellen allgemeiner Kategorien von Benutzern verwendet, die dieselben Sicherheitsrechte in einer Anwendung gemeinsam nutzen. Während sie konzeptuell den .NET-Rollen ähneln, sind sie doch völlig eigenständig.

Enterprise Services (COM+)-Rollen enthalten Windows-Benutzerkonten und -Gruppenkonten (anders als .NET-Rollen, die beliebige Nicht-Windows-Benutzeridentitäten enthalten können). Daher stellen Enterprise Services (COM+)-Rollen nur einen effektiven Autorisierungsmechanismus für Anwendungen dar, die die Windows-Authentifizierung und den Identitätswechsel (um den Sicherheitskontext des Aufrufers an die Enterprise Services-Anwendung zu übermitteln) verwenden.

Tabelle 9.1: *Vergleichen von Enterprise Services (COM+)-Rollen mit .NET-Rollen*

| Feature | Enterprise Services (COM+)-Rollen | .NET-Rollen |
|---|---|---|
| Verwaltung | Verwaltungsprogramm für Komponentendienste | Benutzerdefiniert |
| Datenspeicher | COM+-Katalog | Benutzerdefinierter Datenspeicher (z. B. SQL Server oder Active Directory) |
| Deklarativ | Ja<br>[SecurityRole("Manager")] | Ja<br>[PrincipalPermission(<br>SecurityAction.Demand,<br>Role="Manager")] |
| Imperativ | Ja<br>ContextUtil.IsCallerInRole() | Ja<br>IPrincipal.IsInRole |

| | | |
|---|---|---|
| Feinstufigkeit auf Klassen-, Schnittstellen- und Methodenebene | Ja | Ja |
| Erweiterbar | Nein | Ja<br>(mithilfe einer benutzerdefinierten **IPrincipal**-Implementierung) |
| Verfügbar für alle .NET-Komponenten | Nur für Komponenten, die von der **ServicedComponent**-Basisklasse abstammen. | Ja |
| Rollenmitgliedschaft | Rollen enthalten Windows-Gruppen- oder -Benutzerkonten | Bei Verwendung von **WindowsPrincipals** SIND Rollen Windows-Gruppen (keine zusätzliche Abstraktionsebene). |
| Explizite Schnittstellen-implementierung erforderlich | Ja<br>Für eine Autorisierung auf Methodenebene muss explizit eine Schnittstelle definiert und implementiert werden. | Nein |

# Authentifizierung

Da Enterprise Services von der zugrunde liegenden Struktur abhängen, die von COM+ und DCOM/RPC bereitgestellt wird, handelt es sich bei den für Enterprise Services-Anwendungen verfügbaren Einstellungen der Authentifizierungsebene um die von RPC definierten (und von DCOM verwendeten) Einstellungen.

Tabelle 9.2: *Authentifizierungseinstellungen für Enterprise Services-Anwendungen*

| Authentifizierungsebene | Beschreibung |
|---|---|
| Standard | Auswählen der Authentifizierungsebene unter Verwendung normaler Verhandlungsregeln. |
| Keine | Keine Authentifizierung. |
| Verbinden | Anmeldeinformationen nur authentifizieren, wenn der Client die erste Verbindung zum Server herstellt. |
| Aufruf | Authentifizierung am Anfang jedes Remoteprozeduraufrufs. |
| Paket | Authentifizierung aller vom Client erhaltenen Daten. |
| Paketintegrität | Authentifizierung aller Daten und Überprüfung, dass keine der übertragenen Daten geändert wurden. |
| Paketsicherheit | Authentifizierung aller Daten und Verschlüsselung des Parameterstatus für jeden Remoteprozeduraufruf. |

## Heraufstufen der Authentifizierungsebene

Sie sollten sich bewusst sein, dass bestimmte Authentifizierungsebenen automatisch heraufgestuft werden. Beispiel:

- Bei Verwendung des UDP-Datagrammtransports werden die Ebenen **Verbinden** und **Aufruf** zu **Paket** heraufgestuft, da die zuvor erwähnten Authentifizierungsebenen nur über einen verbindungsorientierten Transport wie TCP sinnvoll sind.

---

**Hinweis:** Windows 2000 verwendet für die DCOM-Kommunikation standardmäßig RPC über TCP.

---

- Für prozessübergreifende Aufrufe auf einem einzelnen Computer werden alle Authentifizierungsebenen immer zur **Paketsicherheit** heraufgestuft. Daten werden in einem Szenario mit einem einzelnen Computer jedoch nicht aus Vertraulichkeitsgründen verschlüsselt (da diese Daten nicht über ein Netzwerk übertragen werden).

### Verhandeln der Authentifizierungsebene

- Die von Enterprise Services zum Authentifizieren eines Clients verwendete Authentifizierungsebene wird durch zwei Einstellungen bestimmt:?
  **Authentifizierungsebene für die Prozessebene** – Bei einer durch Server aktivierten Anwendung (wird innerhalb von **Dllhost.exe** ausgeführt) wird die Authentifizierungsebene innerhalb des COM+-Katalogs konfiguriert.
- **Clientauthentifizierungsebene** – Die konfigurierte Authentifizierungsebene des Clientprozesses, der mit der Serviced Component kommuniziert, hat auch Auswirkungen auf die verwendete Authentifizierungsebene.
  Die Standardauthentifizierungsebene für eine ASP.NET-Webanwendung wird in der Datei **Machine.config** vom **comAuthenticationLevel**-Attribut des Elements <**processModel**> definiert.

Es wird immer die höherwertige der beiden (Client und Server) Authentifizierungsebenen verwendet. Dies wird in Abbildung 9.4 veranschaulicht.

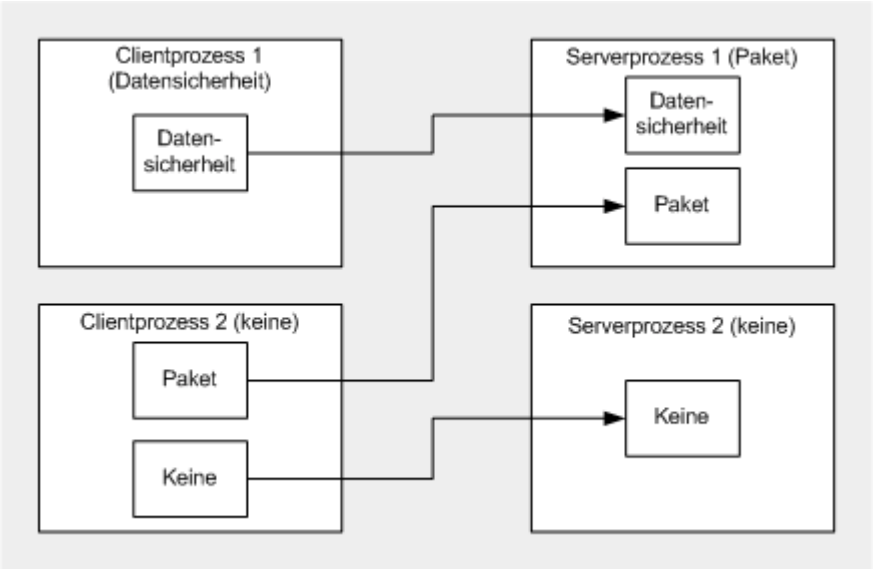


Abbildung 9.4
*Verhandeln der Authentifizierungsebene*

### Weitere Informationen

Weitere Informationen zum Konfigurieren der Authentifizierungsebenen für eine Enterprise Services-Anwendung finden Sie unter "Konfigurieren der Sicherheit" weiter oben in diesem Kapitel.

## Identitätswechsel

Die für eine Enterprise Services-Anwendung definierte Identitätswechselebene bestimmt die für alle ausgehenden DCOM-Aufrufe verwendete Identitätswechselebene, die von Serviced Components innerhalb der Anwendung durchgeführt werden.

**Wichtig:** Sie bestimmt NICHT, ob Serviced Components in der Anwendung den Identitätswechsel für ihre Aufrufer durchführen. Serviced Components führen standardgemäß keinen Identitätswechsel für Aufrufer durch. Damit dies der Fall ist, muss die Serviced Component **CoImpersonateClient** aufrufen, wie unter "Übermitteln des ursprünglichen Aufrufers" weiter oben in diesem Kapitel beschrieben.

Der Identitätswechsel ist eine clientseitige Einstellung. Er bietet dem Client einen gewissen Grad an Schutz, da der Client damit die Möglichkeiten des Servers für den Identitätswechsel einschränken kann.

Tabelle 9.3: *Verfügbare Identitätswechselebenen*

| Identitätswechselebene | Beschreibung |
|---|---|
| **Identify** | Ermöglicht es dem Server, den Client zu identifizieren sowie Zugriffsüberprüfungen unter Verwendung des Zugriffstokens des Clients durchzuführen. |
| **Impersonate** | Ermöglicht es dem Server, unter Verwendung der Anmeldeinformationen des Clients auf lokale Ressourcen zuzugreifen. |
| **Delegate** | Ermöglicht es dem Server, unter Verwendung der Anmeldeinformationen des Clients auf Remoteressourcen zuzugreifen (dies erfordert Kerberos und eine bestimmte Kontokonfiguration). |

Die Standardebene für den Identitätswechsel, die von einer webbasierten Anwendung zum Kommunizieren mit Serviced Components verwendet wird (oder einer beliebigen DCOM verwendenden Komponente), wird in der Datei **Machine.config** durch das **comImpersonationLevel**-Attribut des Elements <**processModel**> bestimmt.

## Cloaking

Das Cloaking bestimmt genau, wie die Clientidentität während des Identitätswechsels über einen COM-Objektproxy für einen Server dargestellt wird. Es gibt zwei Arten von Cloaking:

- **Dynamisches Cloaking** – Enterprise Services-Serveranwendungen verwenden das dynamische Cloaking (ist nicht konfigurierbar). Für Bibliotheksanwendungen wird das Cloaking durch den Hostprozess bestimmt, z. B. durch den ASP.NET-Workerprozess (**Aspnet_wp.exe**). Webbasierte Anwendungen verwenden ebenfalls das dynamische Cloaking (auch dies ist nicht konfigurierbar).

  Das dynamische Cloaking führt dazu, dass das Identitätswechseltoken für den Thread während des Identitätswechsels zum Darstellen der Clientidentität verwendet wird. Das bedeutet, dass beim Aufrufen von **CoImpersonateClient** innerhalb einer Serviced Component die Identität des Clients für nachfolgende ausgehende Aufrufe angenommen wird, die von derselben Methode durchgeführt werden, bis entweder **CoRevertToSelf** aufgerufen wird oder die Methode endet (wobei **CoRevertToSelf** implizit aufgerufen wird).
- **Statisches Cloaking** – Beim statischen Cloaking werden dem Server die Anmeldeinformationen angezeigt, die beim ersten Aufruf des Clients an den Server verwendet werden (unabhängig davon, ob ein Thread während eines ausgehenden Aufrufs einen Identitätswechsel durchführt).

### Weitere Informationen

- Weitere Informationen zum Konfigurieren der Identitätswechselebenen für Enterprise Services-Anwendungen finden Sie unter "Konfigurieren der Sicherheit" weiter oben in diesem Kapitel.
- Weitere Informationen zum Cloaking finden Sie in MSDN bei den Informationen für das Plattform-SDK unter "Cloaking" (englischsprachig).

# Zusammenfassung

In diesem Kapitel wurde beschrieben, wie sichere Serviced Components innerhalb einer Enterprise Services-Anwendung erstellt werden. Es wurde außerdem gezeigt, wie eine webbasierte ASP.NET-Clientanwendung konfiguriert wird, die Serviced Components aufruft. Zusammenfassend kann festgestellt werden:

- Verwenden Sie durch Server aktivierte Enterprise Services-Anwendungen, um die Sicherheit zu erhöhen. Zusätzliche Prozesshops erhöhen ebenfalls die Sicherheit.
- Verwenden Sie zum Ausführen von Serveranwendungen lokale Konten mit minimalen Rechten.
- Verwenden Sie die Authentifizierung auf Paketsicherheitsebene (die auf dem Server und dem Client konfiguriert werden muss), wenn Sie Daten sichern müssen, die zwischen einer Serviced Component und einer Clientanwendung über ein Netzwerk übertragen werden.
- Aktivieren Sie die Zugriffsüberprüfungen auf Komponentenebene, um eine sinnvolle, rollenbasierte Sicherheitsimplementierung zu erreichen.
- Verwenden Sie in einer ASP.NET-Webanwendung die Windows-Authentifizierung, und aktivieren Sie den Identitätswechsel, bevor Sie eine Komponente in einer Enterprise Services-Anwendung aufrufen, die von der rollenbasierten Sicherheit abhängt.
- Verwenden Sie gesicherte Gatewayklassen als Einstiegspunkte in Enterprise Services-Anwendungen.

  Indem Sie die Anzahl der Gatewayklassen verringern, die Einstiegspunkte für Clients in Ihren Enterprise Services-Anwendungen bereitstellen, reduzieren Sie die Anzahl der Klassen, denen Rollen zugeordnet werden müssen. Für andere interne Hilfsklassen sollten rollenbasierte Überprüfungen aktiviert, diesen aber keine Rollen zugeordnet werden. Das bedeutet, dass externe Clients diese nicht direkt aufrufen können, während die Gatewayklassen in derselben Anwendung über den direkten Zugriff verfügen.
- Rufen Sie **IsSecurityEnabled** unmittelbar vor dem Überprüfen der Rollenmitgliedschaft programmatisch auf.
- Vermeiden Sie einen Identitätswechsel auf der mittleren Ebene, da dies die effektive Verwendung des Datenbankverbindungspoolings verhindert und die Skalierbarkeit Ihrer Anwendung drastisch verringert.
- Fügen Sie Windows-Gruppen zu Enterprise Services (COM+)-Rollen hinzu, um die Flexibilität zu erhöhen und die Verwaltung zu vereinfachen.