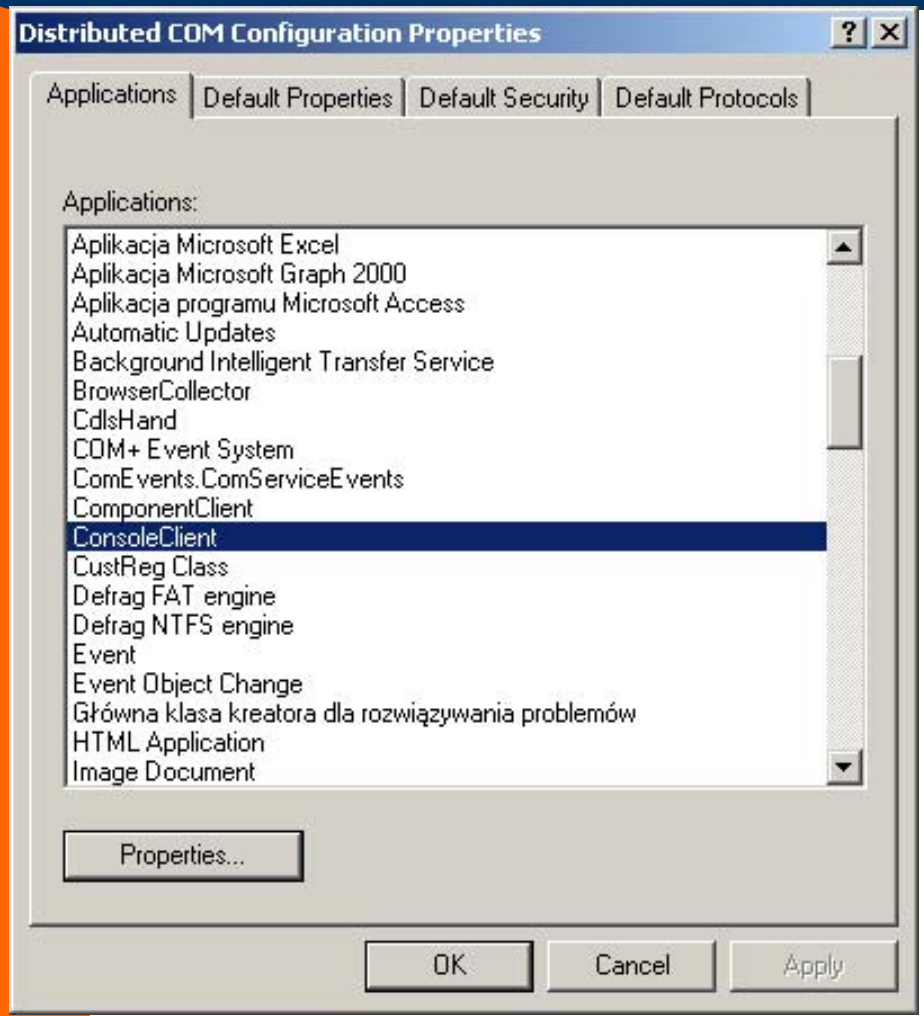
Zdalne uruchamianie obiektów COM

Zdalne uruchamianie obiektów COM

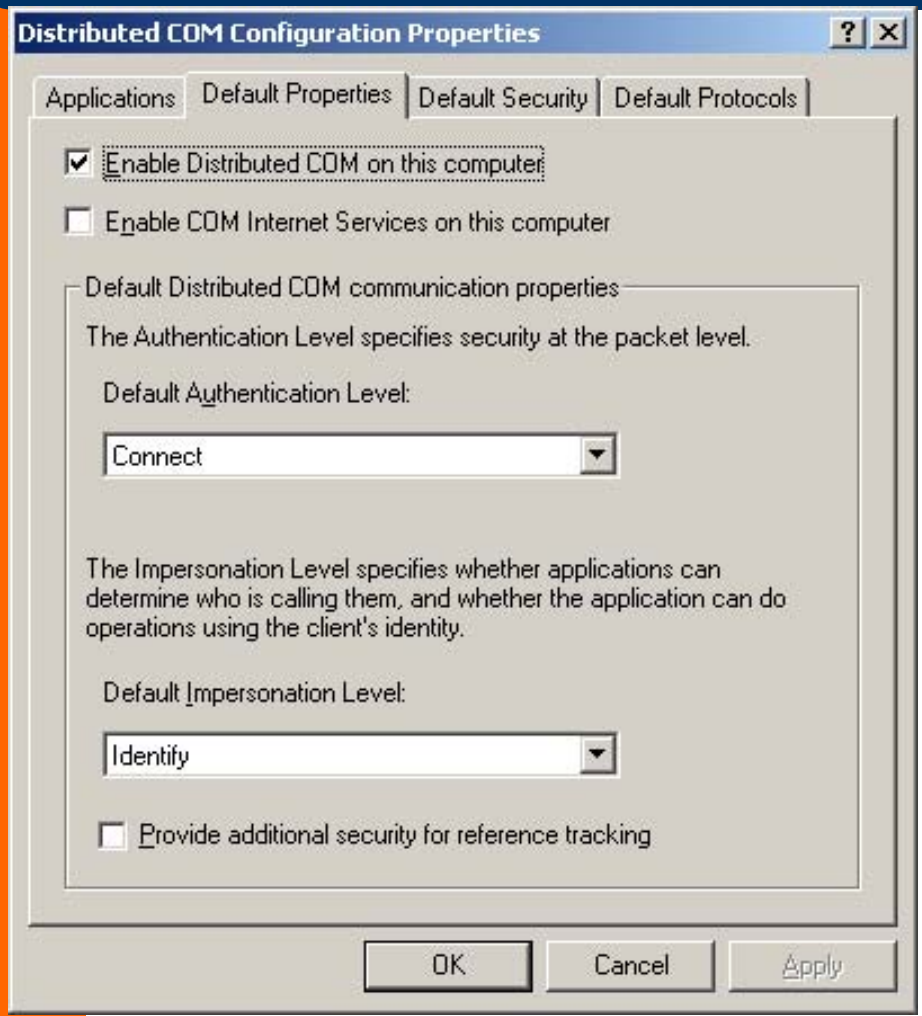
Obiekty COM nie wymagają szczególnej adaptacji, aby mogły być uruchamiane zdalnie. Komponent musi być zarejestrowany w systemie klienta jak i zdalnym. Do zdalnego dostępu do obiektów używany jest protokół RPC, do którego bezpośrednio odwołuje się kod *marshalera.* Wymaga to więc, w przypadku użycia danych niekompatybilnych z `oleautomation`, przygotowania komponentu *proxy/stub* i jego rejestracji w systemach klienta i zdalnej maszynie (`oleautomation` używa biblioteki `ole32.dll`, która jest standardowym elementem systemu Windows). Konfiguracja praw dostępu podsystemu COM musi zezwalać na zdalny dostęp do komponentów przez użytkownika klienta.

Zdalne uruchamianie obiektów COM



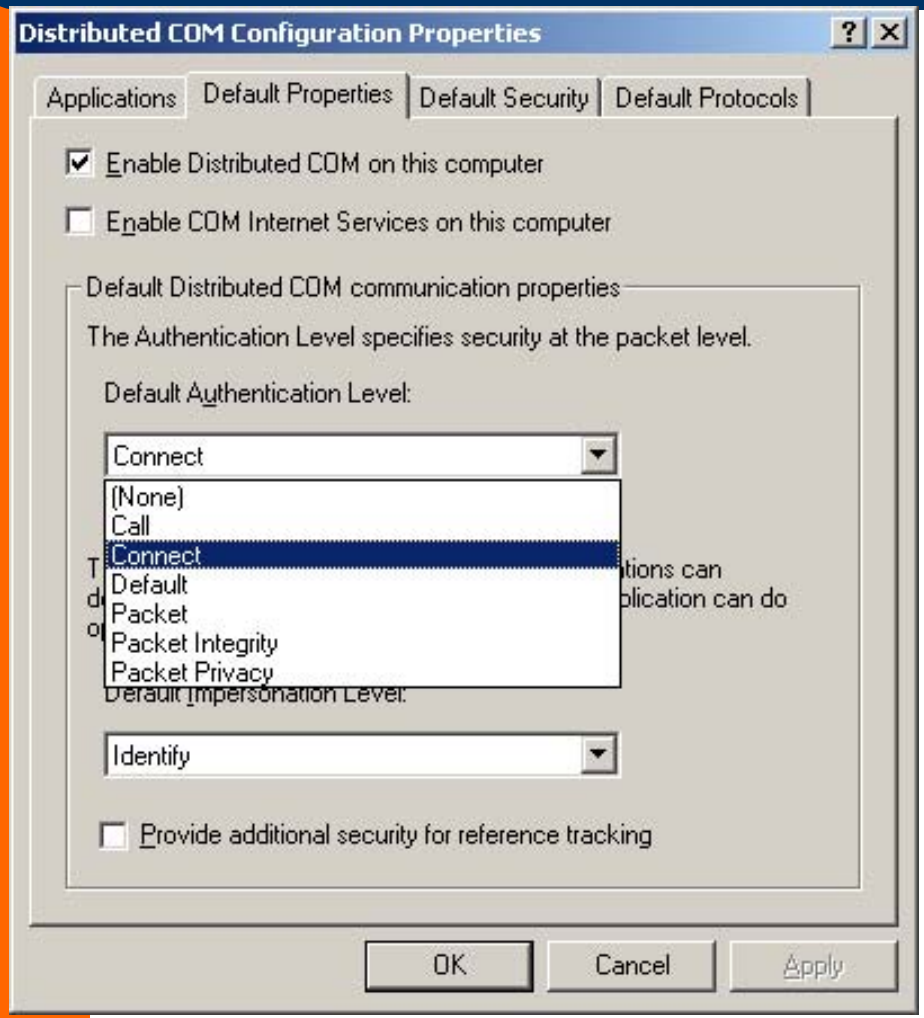
Do definiowania konfiguracji podsytemu COM służy program `DCOMCnfg.exe`. Pozwala on na określenie systemowych praw dostępu do komponentów, jak również sposobu uruchomienia i praw dostępu dla poszczególnych komponentów.

Zdalne uruchamianie obiektów COM



Defaut Properties **definiują domyślną konfigurację dla wszystkich komponentów.**
Opcja *Enable Distributed COM* pozwala na udostępnianie komponentów zdalnym klientom.
Opcja *Enable COM Internet Services* pozwala na dostęp do komponentów za pomocą protokołu HTTP. Pozwala to na współpracę komponentów znajdujących się za *firewall*, które najczęściej nie filtrują tego protokołu.

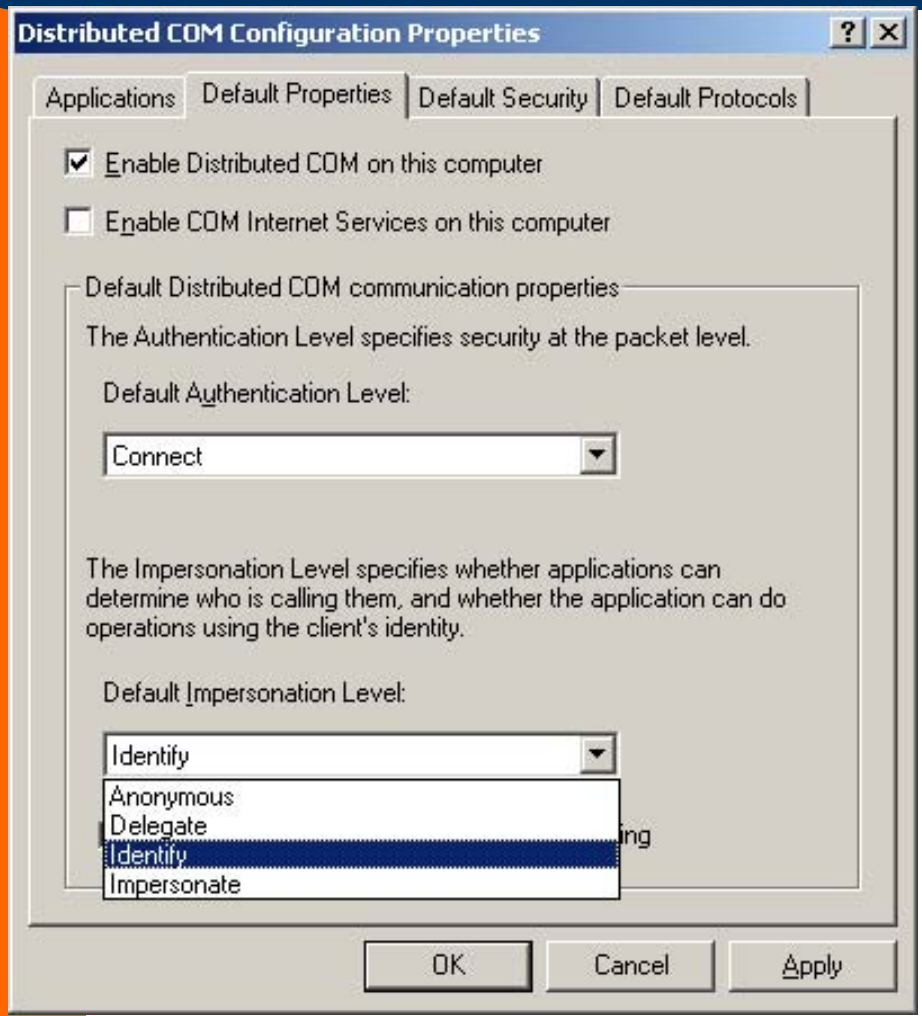
Zdalne uruchamianie obiektów COM



Defaut Autentication Level **określa sposób autoryzacji podczas komunikacji:**

- ***Connect (Default)*** **- autoryzacja następuje podczas pierwszego połączenia się z komponentem.**
- ***Call*** **- autoryzacja następuje podczas każdego zdalnego wywołania.**
- ***Packet*** **- autoryzacja wszystkich przesyłanych danych (pakietów).**
- ***Packet Integrity*** **- jak *Packet*, dodatkowo dane są sprawdzane czy nie nastąpiła ich modyfikacja podczas transferu.**
- ***Packet Privacy*** **- jak *Packet Integrity*, dane są dodatkowo szyfrowane.**

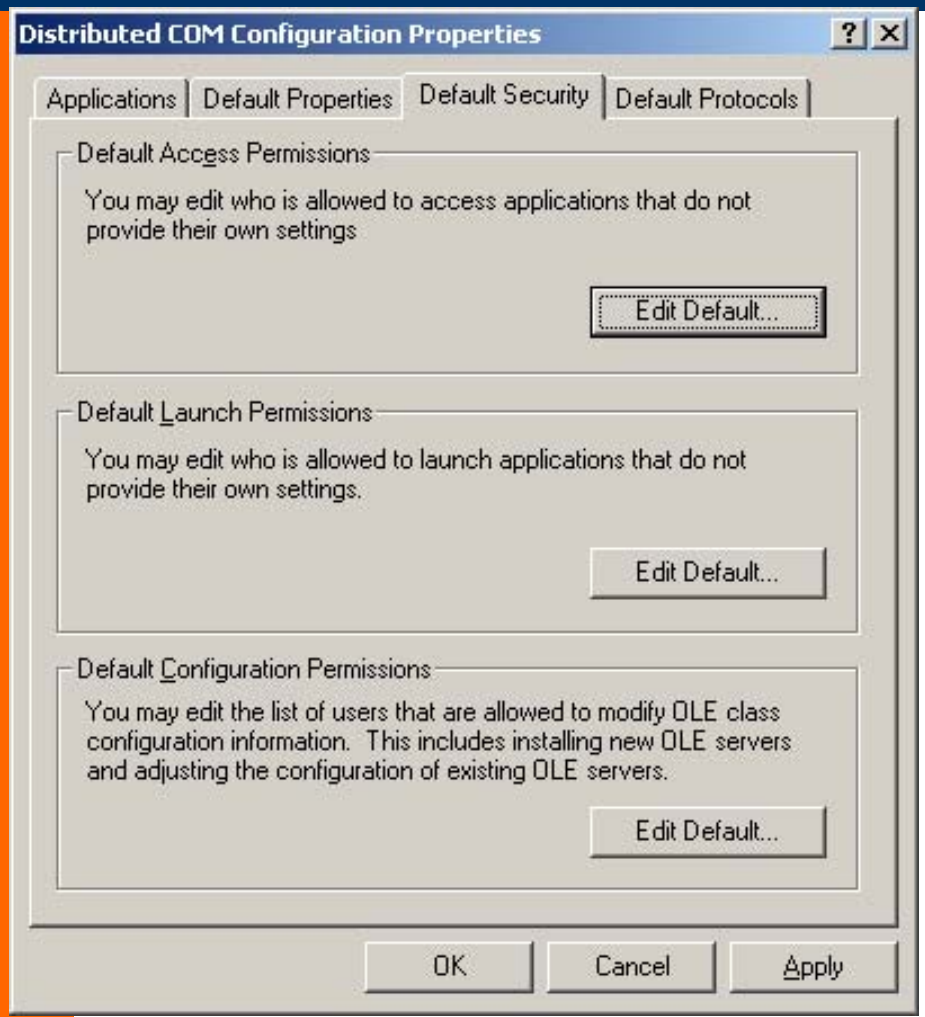
Zdalne uruchamianie obiektów COM



Defaut Impersonation Level **definiuje prawa jakie zyskuje komponent klient uruchamiany przez klienta:**

- ***Anonymous*** **- klient pozostaje nieznany serwerowi. Żadne akcje w systemie nie mogą zostać wykonane na rzecz klienta.**
- ***Identify*** **- serwer może otrzymać dane identyfikujące klienta, prawa pozostają jak dla *Anonymous*.**
- ***Impersonate*** **- serwer może otrzymać dane identyfikujące klienta i operować na systemie na jego rzecz.**
- ***Delegate*** **- jak *Impersonate*, dodatkowo serwer może wywoływać obiekty innych serwerów na rzecz klienta.**

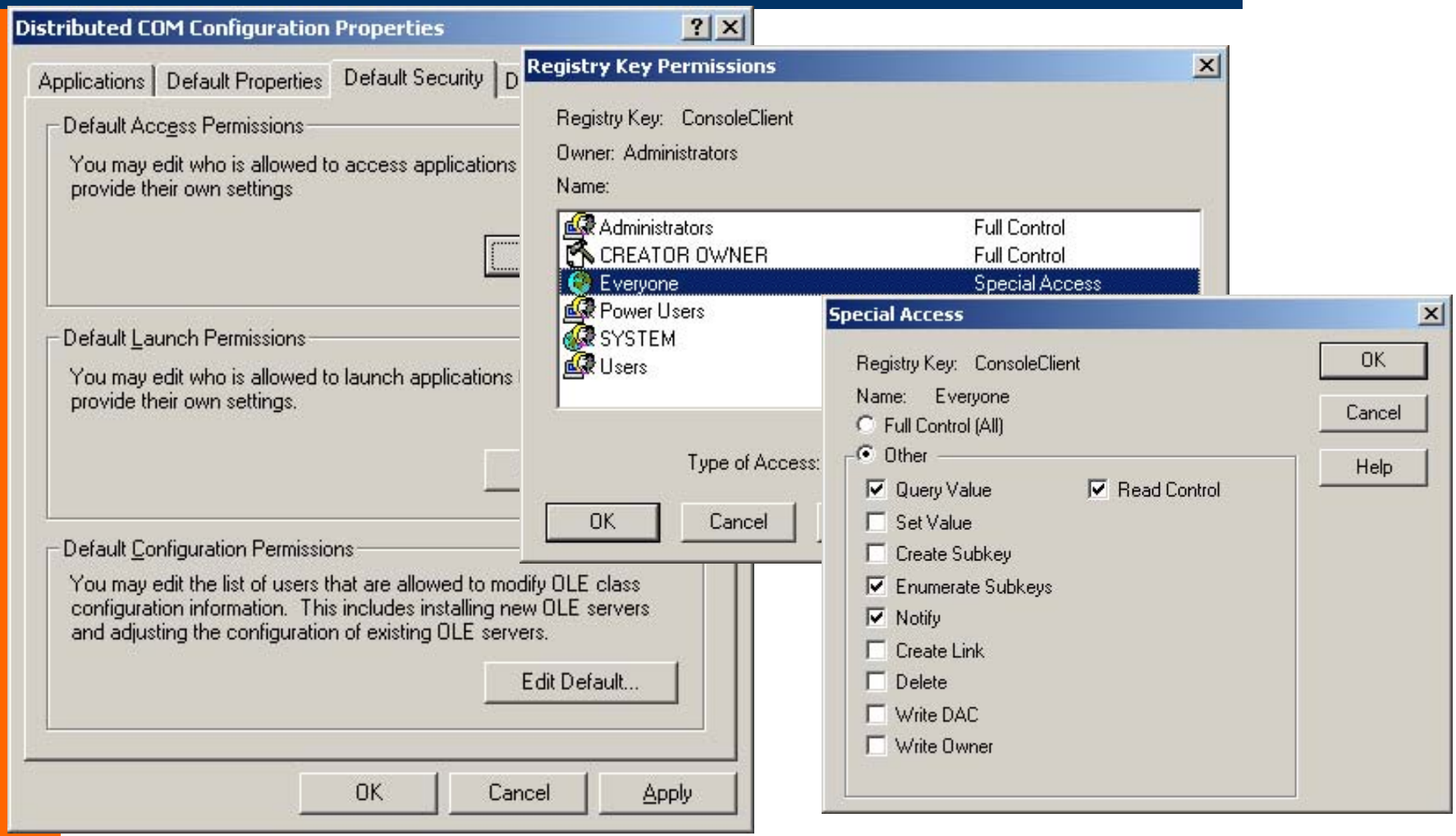
Zdalne uruchamianie obiektów COM



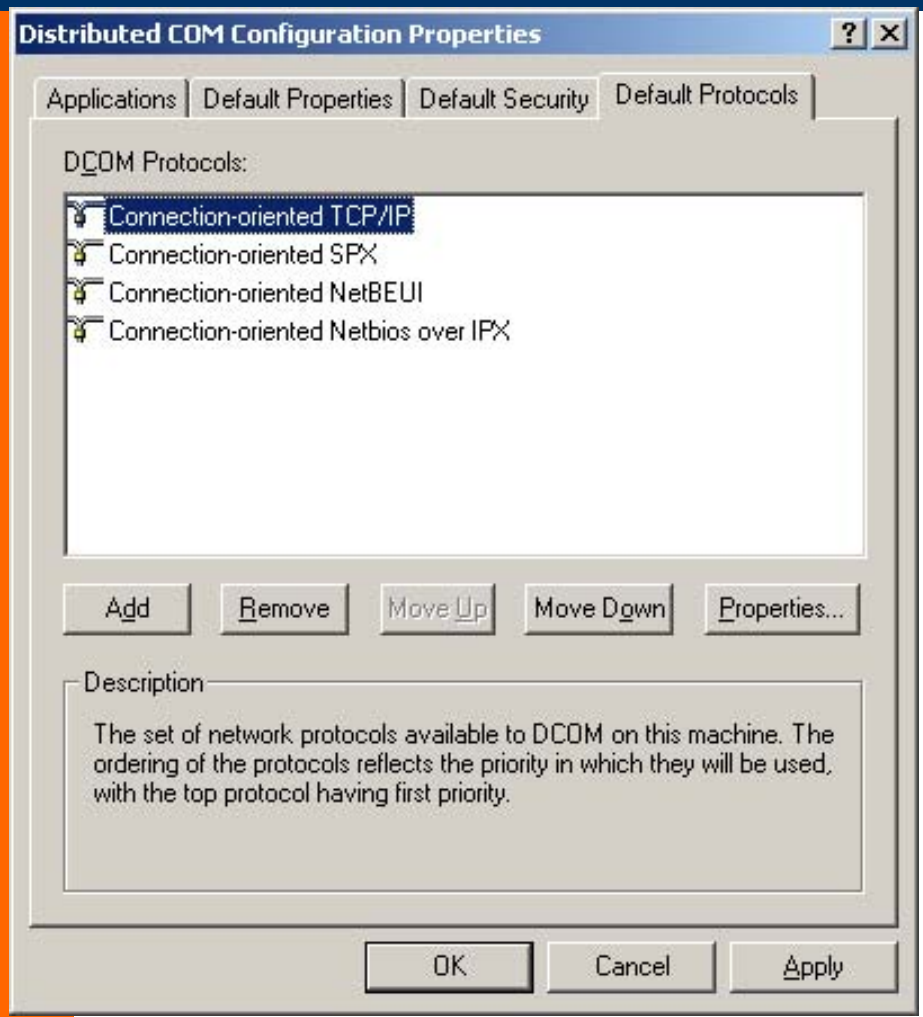
Defaut Security **określa prawa użytkowników do uruchamiania i dostępu do obiektów COM:**

- ***Default Launch Permissions*** **określają użytkowników, którzy mogą uruchomić komponent.**
- ***Default Access Permissions*** **określają użytkowników, którzy mają dostęp do komponentu.**
- ***Default Configuration Permissions*** **definiuje użytkowników, którzy mogą modyfikować prawa dostępu do komponentów.**

Zdalne uruchamianie obiektów COM

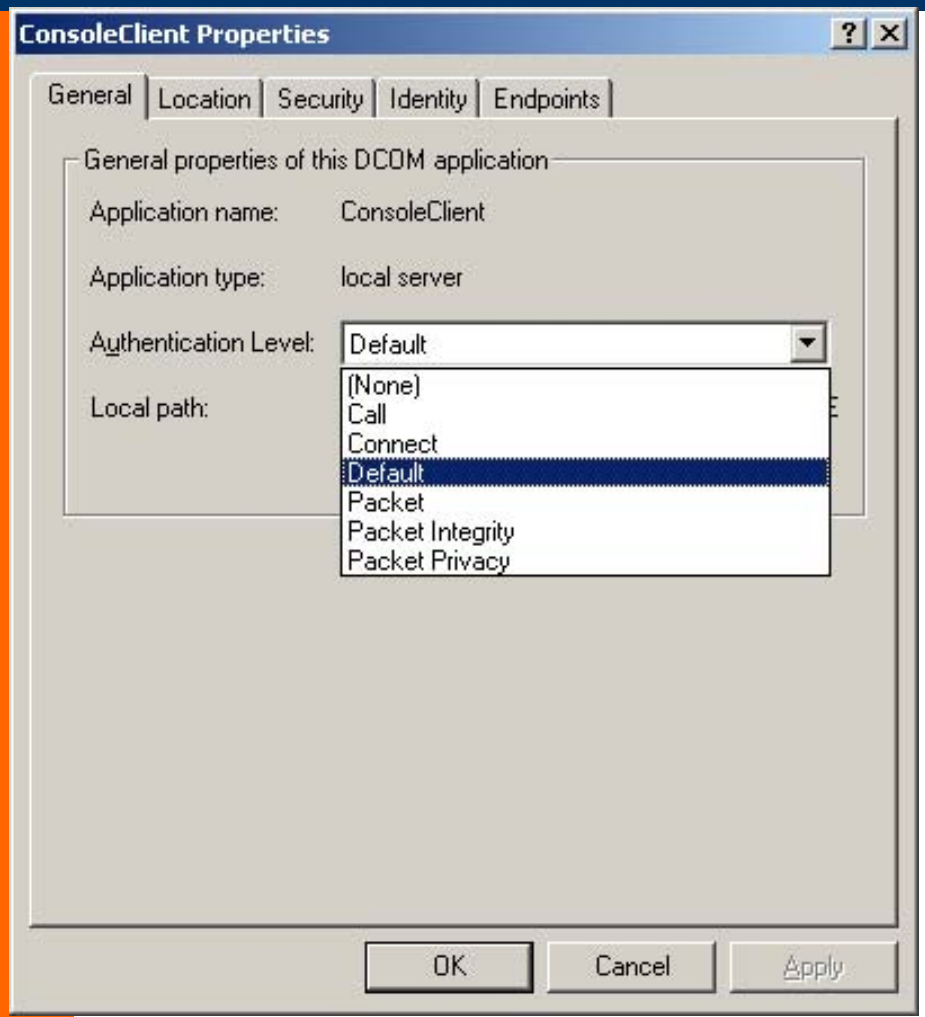


Zdalne uruchamianie obiektów COM



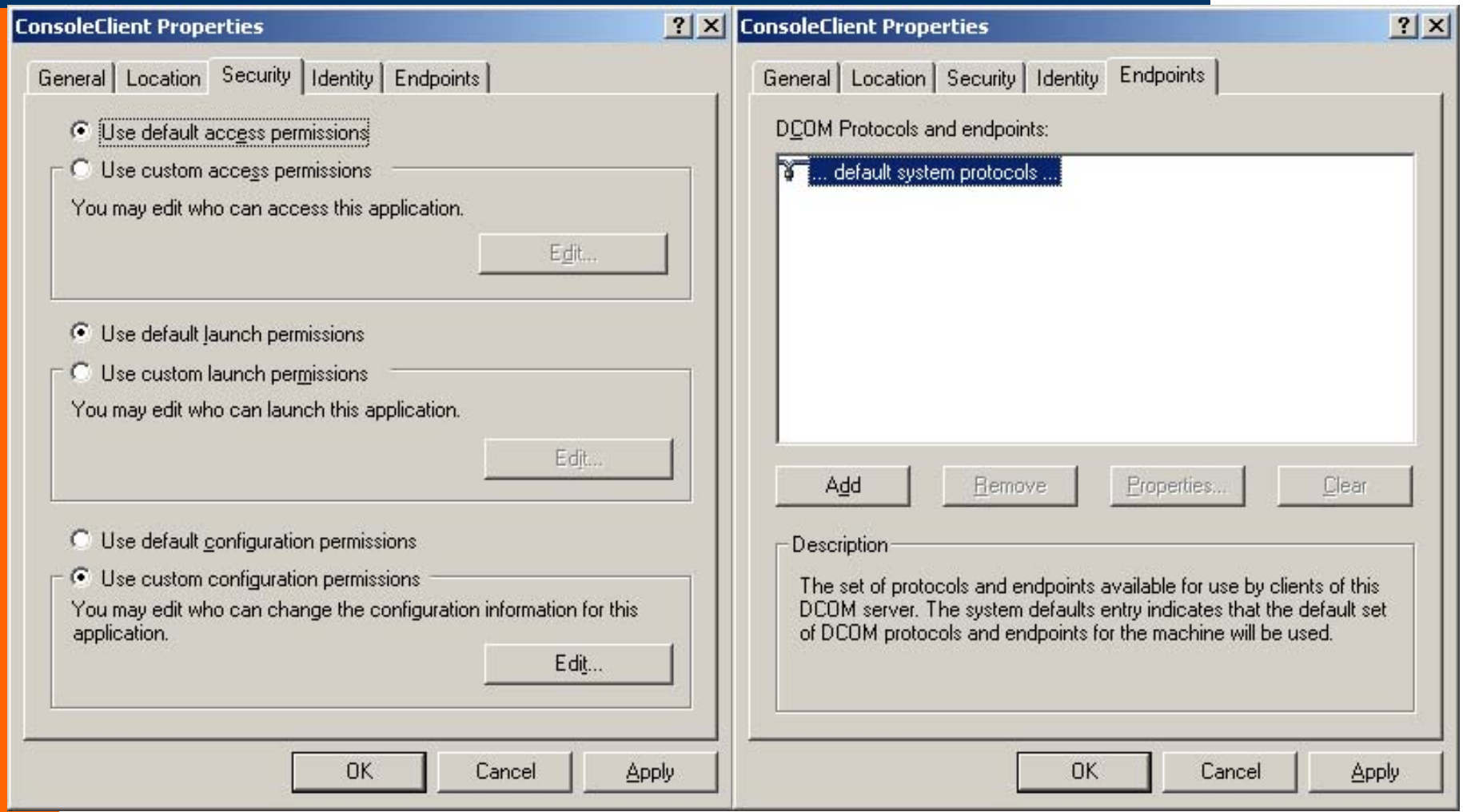
Defaut Protocols **określa które z protokołów mogą być użyte do komunikacji z serwerem. Dodatkowo definiowana jest kolejność, wg której będą podejmowane próby połączenia.**

Zdalne uruchamianie obiektów COM

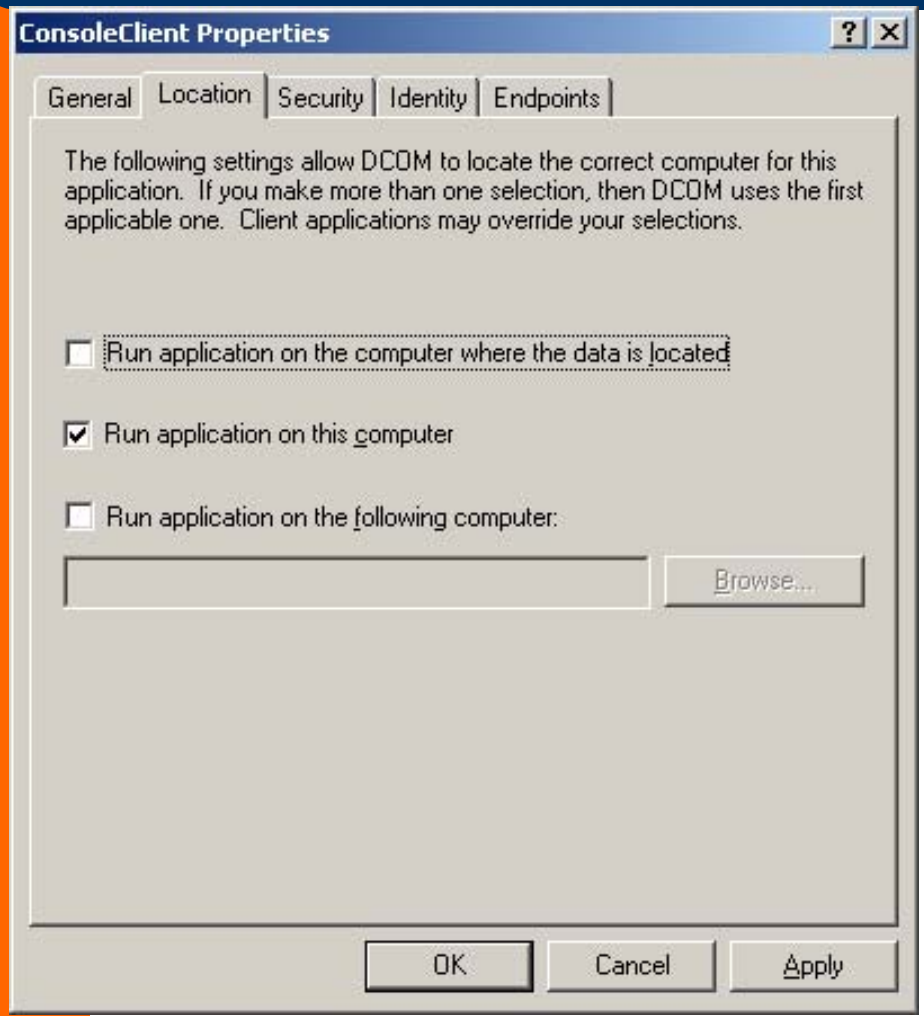


Domyślna konfiguracja może być modyfikowana indywidualnie dla każdego komponentu. Jedynie te specyficzne ustawienia są ważne, które zostały zdefiniowane. Dla pozostałych uwzględniane są ustawienia domyślne.

Zdalne uruchamianie obiektów COM



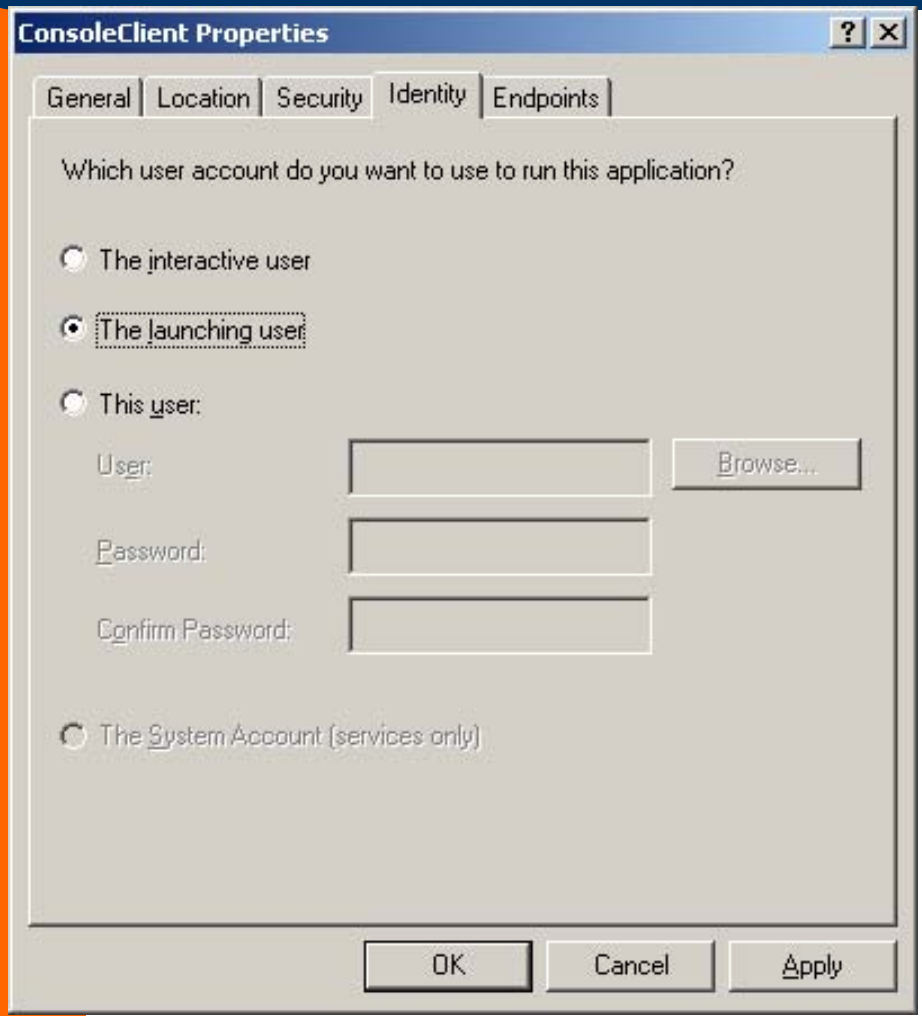
Zdalne uruchamianie obiektów COM



Location **definiuje domyślny komputer, na którym ma zostać uruchomiony komponent:**

- ***Run application on the computer where the data is located*** **uruchamia komponent na maszynie, na której znajduje się zapisany stan obiektu.**
- ***Run application on this computer*** **uruchamia komponent lokalnie.**
- ***Run application on the following computer*** **jawnie specyfikuje nazwę komputera, na którym na zostać uruchomiony komponent (na zdefiniowanym komputerze komponent musi zostać zdefiniowany jako *Run application on this computer*).**

Zdalne uruchamianie obiektów COM



Identity **definiuje użytkownika, który zostanie użyty do uruchomienia komponentu:**

- ***The interactive user*** **- użytkownik aktualnie zalogowany. Jeśli żaden użytkownik nie jest zalogowany, uruchomienie komponentu się nie powiedzie.**
- ***The launching user*** **- użytkownik uruchamiający komponent (skojarzony z aplikacją klienta).**
- ***This user*** **- wskazany jawnie użytkownik.**

Zdalne uruchamianie obiektów COM

Zdefiniowane w rejestrze parametry dotyczące sposobu aktywacji komponentu dotyczą domyślnego sposobu jego uruchomienia poprzez funkcję `CoCreateInstance()`. Można je również wyspecyfikować z poziomu kodu klienta poprzez użycie metody `CoCreateInstanceEx()`:

```
HRESULT CoCreateInstanceEx(
    REFCLSID rclsid,           //CLSID of the object to be created
    IUnknown *punkOuter,       //If part of an aggregate, the
                               // controlling IUnknown
    DWORD dwClsCtx,            //CLSCTX values
    COSERVERINFO *pSrvInfo,    //Machine on which the object is to
                               // be instantiated
    ULONG cmq,                 //Number of MULTI_QI structures in
                               // pResults
    MULTI_QI *pResults         //Array of MULTI_QI structures
);
```

Zdalne uruchamianie obiektów COM

Pozwala ona na wyspecyfikowanie w strukturze komputera na którym ma zostać uruchomiony komponent:

```
typedef struct  _COSERVERINFO
{
    DWORD dwReserved1;
    LPWSTR pwszName;
    COAUTHINFO  *pAuthInfo;
    DWORD dwReserved2;
}   COSERVERINFO;
```

Dodatkowo poprzez strukturę `MULTI_QI` można zażądać wielu wskaźników do interfejsów jednocześnie eliminując konieczność wielokrotnych zapytań przez `QueryInterface()`.

```
typedef struct _MULTI_QI
{
        const IID*    pIID;
        IUnknown *    pItf;
        HRESULT       hr;
} MULTI_QI;
```

Kontrola praw dostępu

Aby uniezależnić się od typowego dla Windows NT systemu zabezpieczeń, system COM korzysta z SSPI (*Security Service Provider Interface*). SSPI jest niezależnym od platformy API opracowanym do zarządzania atrybutami praw dostępu. Windows NT wspiera *NT LAN Manager Security Support Provider* (NTLM SSP), począwszy od Windows 2000 dostępny jest również *Kerberos Security Provider*. Oba SSPI są wykorzystywane na poziomie RPC.

Do manipulacji na informacjach o zabezpieczeniach, Microsoft opracował interfejs `IAccessControl`. Standardowa implementacja tego interfejsu znajduje się w komponencie `CLSID_DCOMAccessControl`. Komponent ten dodatkowo implementuje interfejsy IPersist i IPersistStream, co pozwala na łatwe operacje zapisu i odczytu ustawień z plików lub rejestrów systemowych.

Kontrola praw dostępu

```
interface IAccessControl : IUnknown
{
  HRESULT GrantAccessRights ( [in] PACTRL_ACCESS pAccessList );

  HRESULT SetAccessRights ( [in] PACTRL_ACCESS pAccessList );

  HRESULT SetOwner ( [in] PTRUSTEE pOwner, [in] PTRUSTEEW pGroup );

  HRESULT RevokeAccessRights ( [in] LPWSTR lpProperty,
    [in] ULONG cTrustees, [in, size_is(cTrustees)] TRUSTEE prgTrustees[] );

  HRESULT GetAllAccessRights ( [in] LPWSTR lpProperty,
    [out] PACTRL_ACCESSW_ALLOCATE_ALL_NODES *ppAccessList,
    [out] PTRUSTEE *ppOwner, [out] PTRUSTEE *ppGroup );

  HRESULT IsAccessAllowed ( [in] PTRUSTEE pTrustee,
    [in] LPWSTR lpProperty, [in] ACCESS_RIGHTS AccessRights,
    [out] BOOL *pfAccessAllowed );
}
```

Kontrola praw dostępu

```
typedef struct {
    ULONG                       cEntries;
    PACTRL_PROPERTY_ENTRY       pPropertyAccessList;
} ACTRL_ACCESS;
typedef struct {
    LPCTSTR                        lpProperty;
    PACTRL_ACCESS_ENTRY_LIST       pAccessEntryList;
    ULONG                          fListFlags;
} ACTRL_PROPERTY_ENTRY;
typedef struct {
    ULONG                             cEntries;
    PACTRL_ACCESS_ENTRY               pAccessList;
} ACTRL_ACCESS_ENTRY_LIST;
typedef struct {
    TRUSTEE                     Trustee;
    ULONG                       fAccessFlags;
    ACCESS_RIGHTS               Access;
    ACCESS_RIGHTS               ProvSpecificAccess;
    INHERIT_FLAGS               Inheritance;
    LPCTSTR                     lpInheritProperty;
} ACTRL_ACCESS_ENTRY;
typedef struct {
  PTRUSTEE                         pMultipleTrustee;
  MULTIPLE_TRUSTEE_OPERATION       MultipleTrusteeOperation;
  TRUSTEE_FORM                     TrusteeForm;
  TRUSTEE_TYPE                     TrusteeType;
  LPTSTR                           ptstrName;
} TRUSTEE;
```

Kontrola praw dostępu

`GrantAccessRights()` uzupełnia istniejącą tablicę uprawnień o nowe wpisy.

`SetAccessRights()` ustawia nową tablicę uprawnień.

`RevokeAccessRights()` usuwa uprawnienia wskazanych użytkowników z listy.

`GetAllAccessRights()` odczytuje całą listę uprawnień pamiętaną przez obiekt.

`IsAccessAllowed()` sprawdza czy wskazany rodzaj operacji jest dozwolony użytkownikowi.

`SetOwner()` ustawia nowego właściciela obiektu uprawnień (nie zaimplementowane w systemowym komponencie. Właścicielem jest zawsze użytkownik tworzący obiekt).

Kontrola praw dostępu z poziomu kodu komponentu

Komponent może ustalać prawa dostępu na drodze programowej. Pozwala to na wprowadzenie dodatkowych ograniczeń dostępu do komponentu, np. ograniczyć dostęp do wybranych metod. Inicjalizacja praw dostępu następuje poprzez wywołanie `CoInitializeSecurity()`. Wywołanie to powinno wystąpić przed pierwszym przesłaniem interfejsu do klienta. W przeciwnym razie inicjalizacja przeprowadzana jest automatycznie z domyślnymi ustawieniami uprawnień dla komponentu. Uprawnienia są wspólne dla całego procesu, tak więc `CoInitializeSecurity()` powinno być wywołane jak najwcześniej i tylko jeden raz. Kolejne wywołania tej funkcji zwracają błąd `RPC_E_TOO_LATE`.

Kontrola praw dostępu z poziomu kodu komponentu

```
HRESULT CoInitializeSecurity(
 PSECURITY_DESCRIPTOR pVoid,    //Points to security descriptor
 LONG cAuthSvc,                 //Count of entries in asAuthSvc
 SOLE_AUTHENTICATION_SERVICE * asAuthSvc,
                                //Array of names to register
 void * pReserved1,             //Reserved for future use
 DWORD dwAuthnLevel,            //The default authentication level
                                // for proxies
 DWORD dwImpLevel,              //The default impersonation level
                                // for proxies
 SOLE_AUTHENTICATION_LIST * pAuthList,
                                //Authentication information for
                                // each authentication service
 DWORD dwCapabilities,          //Additional client and/or
                                // server-side capabilities
 void * pReserved3              //Reserved for future use
);
```

Kontrola praw dostępu z poziomu kodu komponentu

`PSECURITY_DESCRIPTOR pVoid` - wskazuje na obiekt przechowujący dane o prawach dostępu. Może to być obiekt typu `IAccessControl`, `SECURITY_DESCRIPTOR` (struktura definiująca prawa dostępu dla systemu Windows) lub `GUID` (wskazujące na *{AppID}* w rejestrze, z którego pobrane zostaną prawa dostępu).

Typ obiektu definiowany jest w `DWORD dwCapabilities`:

- **`EOAC_ACCESS_CONTROL` - `IAccessControl`**
- **`EOAC_APPID` - `GUID`**
- **żaden z powyższych - `SECURITY_DESCRIPTOR`**

Najczęściej używanymi dodatkowymi opcjami są:

- **`EOAC_NONE` - brak opcji**
- **`EOAC_SECURE_REFS` - niezależnie od ustawień autoryzacji przeprowadzana jest weryfikacja użytkownika dla wywołań `AddRef()` i `Release()`**

Kontrola praw dostępu z poziomu kodu komponentu

`LONG cAuthSvc` - definiuje liczbę serwisów autoryzacji rejestrowanych w `asAuthSvc`. `-1` oznacza, że podsystem COM wybierze właściwy wśród dostępnych w systemie.

`SOLE_AUTHENTICATION_SERVICE * asAuthSvc` definiuje tablicę serwisów autoryzacji jakie RPC może użyć (Kerberos, WINNT).

`DWORD dwAuthnLevel` - definiuje poziom autoryzacji:

```
RPC_C_AUTHN_LEVEL_DEFAULT
 RPC_C_AUTHN_LEVEL_NONE
 RPC_C_AUTHN_LEVEL_CONNECT
 RPC_C_AUTHN_LEVEL_CALL
 RPC_C_AUTHN_LEVEL_PKT
 RPC_C_AUTHN_LEVEL_PKT_INTEGRITY
 RPC_C_AUTHN_LEVEL_PKT_PRIVACY
```

Kontrola praw dostępu z poziomu kodu komponentu

`DWORD dwImpLevel` - **definiuje sposób identyfikacji klienta:**

```
RPC_C_IMP_LEVEL_DEFAULT
RPC_C_IMP_LEVEL_ANONYMOUS
RPC_C_IMP_LEVEL_IDENTIFY
RPC_C_IMP_LEVEL_IMPERSONATE
RPC_C_IMP_LEVEL_DELEGATE
```

`SOLE_AUTHENTICATION_LIST * pAuthList` - **definiuje tablicę serwisów autoryzacji jakie klient może użyć**

Kontrola praw dostępu z poziomu kodu komponentu

Komponent może próbować identyfikować użytkownika aplikacji klienta w każdej metodzie interfejsu i zezwalać na wykonanie metody w zależności od uprawnień jakie zdefiniujemy dla tego użytkownika. Wyjątek stanowi metoda `QueryInterface()`, która nigdy nie powinna ograniczać dostępu. Do identyfikacji klienta służy interfejs `IServerSecurity`. Wskaźnik do tego interfejsu można uzyskać wewnątrz metody implementującej interfejs poprzez `CoGetCallContext()`:

```
IServerSecurity* pServerSecurity;
HRESULT hr=CoGetCallContext(IID_ IServerSecurity,
                (void**)&pServerSecurity);
```

Kontrola praw dostępu
z poziomu kodu komponentu

```
interface IServerSecurity : IUnknown
{
    HRESULT QueryBlanket (
        [out] DWORD    *pAuthnSvc,
        [out] DWORD    *pAuthzSvc,
        [out] OLECHAR **pServerPrincName,
        [out] DWORD    *pAuthnLevel,
        [out] DWORD    *pImpLevel,
        [out] void     **pPrivs,
        [out] DWORD    *pCapabilities );

    HRESULT ImpersonateClient();

    HRESULT RevertToSelf();

    BOOL IsImpersonating();
}
```

Kontrola praw dostępu z poziomu kodu komponentu

`ImpersonateClient()` tymczasowo przypisuje komponentowi prawa klienta wywołującego metodę. W zależności od ustawień *Impersonation Level* metoda zyskuje następujące przywileje:

- `RPC_C_IMP_LEVEL_ANONYMOUS` - klient jest nieznany dla serwera.
- `RPC_C_IMP_LEVEL_IDENTIFY` - metoda uzyskuje uprawnienia klienta, lecz możliwa jest jedynie identyfikacja klienta i jego ustawień zabezpieczeń. Metoda nie uzyskuje uprawnień do podejmowania akcji na rzecz klienta.
- `RPC_C_IMP_LEVEL_IMPERSONATE` - metoda uzyskuje uprawnienia klienta i ma na jego rzecz dostęp do zasobów komputera.
- `RPC_C_IMP_LEVEL_DELEGATE` - metoda uzyskuje uprawnienia klienta i może na jego rzecz odwoływać się do innych komponentów.

`RevertToSelf()` przywraca użytkownika przypisanego do komponentu. Metoda ta musi być zawsze wywołana po `ImpersonateClient()` przed wyjściem z metody.

Kontrola praw dostępu
Strona klienta

Klient może modyfikować poziom zabezpieczeń połączenia za pomocą interfejsu `IClientSecurity`. Umożliwia to na tymczasowe lub stałe zwiększenie bezpieczeństwa połączenia, np. gdy przesyłane są osobiste dane, które powinny być szyfrowane. Interfejs ten jest implementowany w klasie *proxy* komponentu. Dla komponentów w pliku dll ładowanych w przestrzeń adresową klienta, gdy *proxy* nie jest używane, interfejs ten jest nie jest dostępny (nie jest on potrzebny, bo komponent ma prawa klienta i nie ma problemów bezpieczeństwa podczas wywołań, bo nie jest wykorzystywane RPC - wywołania mają charakter bezpośredni). Dostęp do `IClientSecurity` uzyskuje się `QueryInterface()` na rzecz interfejsu komponentu (a w rzeczywistości *proxy*).

```
IClientSecurity* pClientSecurity;
pComponent->QueryInterface(IID_IClientSecurity,
       (void**)&pClientSecurity);
```

Kontrola praw dostępu
Strona klienta

```
interface IClientSecurity : IUnknown {
  HRESULT QueryBlanket (
        [in]  IUnknown              *pProxy,
        [out] DWORD                 *pAuthnSvc,
        [out] DWORD                 *pAuthzSvc,
        [out] OLECHAR               **pServerPrincName,
        [out] DWORD                 *pAuthnLevel,
        [out] DWORD                 *pImpLevel,
        [out] void                  **pAuthInfo,
        [out] DWORD                 *pCapabilites );
    HRESULT SetBlanket (
        [in] IUnknown               *pProxy,
        [in] DWORD                   AuthnSvc,
        [in] DWORD                   AuthzSvc,
        [in] OLECHAR                *pServerPrincName,
        [in] DWORD                   AuthnLevel,
        [in] DWORD                   ImpLevel,
        [in] void                   *pAuthInfo,
        [in] DWORD                   Capabilities );
    HRESULT CopyProxy (
        [in]  IUnknown  *pProxy,
        [out] IUnknown **ppCopy );
}
```

Kontrola praw dostępu
Strona klienta

`QueryBlanket()` i `SetBlanket()` pozwalają na odczyt i ustawienie poziomu zabezpieczeń dla wskazanego interfejsu.
Jeśli klient używa w różnych częściach kodu tego samego wskaźnika do interfejsu, to zmiany poziomu zabezpieczeń obejmą wszystkie wywołania. Aby zmiany dotyczyły wybranego połączenia (wywołań z określonej części kodu) można wywołanie `SetBlanket()` poprzedzić `CopyProxy()`, które tworzy nowe połączenie do tej samej instancji komponentu.

Należy pamiętać, że nie można obniżyć poziomu zabezpieczeń poniżej określonego przez komponent.