



Configuring Access Rights to Cndex Server on OPENControl and PrimaLogic

Page 1 of 16

Summary

INTRODUCTION	2
AUTHENTICATION LEVEL.....	2
MODIFYING THE CNDEX CONNECTION MODE	2
<i>Activation rights.....</i>	2
<i>Identity.....</i>	3
WINNBI 3.1 AND SUBSEQUENT VERSIONS	3
REMOTE ACCESS TO CNDEX FROM .NET APPLICATIONS	4
ADAPTING A .NET APPLICATION TO CNDEX	4
<i>C#</i>	4
<i>VB.NET</i>	4
REGISTERING A USER WITH THE NTLM PROTOCOL ON OPENCONTROL AND PRIMALOGIC	6
<i>Remote Desktop on PrimaLogic</i>	6
REGISTERING A USER WITH THE NTLM PROTOCOL ON CEWIN	8
COMPATIBILITY OF .NET APPLICATIONS WITH CNDEX ON 64 BIT OPERATING SYSTEMS	9
CHANGING DCOM AND CNDEX SETTINGS ON THE CLIENT MACHINE.....	11
<i>Modifying DCOM default settings</i>	11
<i>Configuring Cndex on the client system</i>	11

Introduction

This document contains explains how to configure DCOM on Windows XP and Windows 2000 systems in order to enable remote access to the Cndex server. Besides, the document contains useful information for C++ and C#/VB.NET developers.

Authentication Level

A DCOM connection uses the highest authentication level between those requested by the server and the client respectively, that is, if the server requests CONENCT authentication level and the client requests NONE, the handshake is performed with CONNECT authentication level since it is the highest between the two.

A connection to a Windows CE DCOM server is never performed with an authentication level higher than CONNECT. For further details:

<http://msdn2.microsoft.com/en-us/library/ms862115.aspx>

Modifying the Cndex connection mode

When a client connects to a DCOM server, if both parts request authentication level NONE, then the server grants access without verifying the client's credentials. In this case the user running the client program does not need to be registered either in the machine running the DCOM server or as a domain user if the machine running the server is part of a Windows domain. On XTend systems this authentication mode removes the need to register users with NTLMUser.exe.

Starting from da WinNBI 3.1, all WinNBI applications connect to Cndex using authentication level NONE (no authentication requested). The DCOM security protocol is configured at runtime with a call to the CoInitializeSecurity API from CndexLink.dll. The authentication level selected by CoInitializeSecurity has priority over default settings configured via Control Panel or dcomcnfg.exe. The caveat is that CoInitializeSecurity can be called once per application, and once the DCOM security protocol is configured it cannot be changed until the application is closed.

Activation rights

This section does not apply to Windows CE systems.

In order to remotely activate a DCOM server it is necessary to grant the SYSTEM user remote access rights to the server itself. Otherwise the system cannot launch the DCOM server when requested by a remote client.

On Windows XP/2000 systems it is necessary to open Component Services->Computer->My Computer->COM Protection, press the Edit Limits button in the Access Permissions panel and enable remote access rights for the ANONYMOUS LOGON user. If such user does not exist it must be added. The same operation must be performed in the Launch and Execution Rights panel.

Identity

It is necessary to select which credentials the server will use once launched. There are three options:

1. Interactive user

With this option the server uses the credentials of the user which is logged into the machine when the server is launched.

Pros: it is easy to determine the server identity.

Cons: requires that a user actually logs into the machine running the server, otherwise it is not possible for the system to launch the server itself.

2. Launching user

With this option the server uses the credentials of the user that has requested the first access (and then the launch of the server).

Pros: the server receives its credentials directly from the client.

Cons: this option is not valid when used together with the authentication level NONE, because in this case the client does not provide any credentials for the server to use. Besides, with this option the system creates an instance of the server for each remote user requesting access. This behaviour does not comply with Cndex specification.

3. This user

In this case the server credentials must be configured either with dcomcnfg.exe or by software. For software configuration see the DCOMPERM code sample in Microsoft Platform SDK.

Pros: assigning a specific user eases the control over the privileges of the DCOM server.

Cons: the assigned user must already have been configured either in the machine or in the Windows domain the machine belongs to.

WinNBI 3.1 and subsequent versions

Cndex configuration for WinNBI 3.1 and subsequent versions:

Authentication level: NONE (no authentication).

This option is selected via CoInitializeSecurity by both the Cndex server and the WinNBI clients, but can be also set as default from Control Panel. Application that do not use CndexLink.dll must call CoInitializeSecurity passing RPC_C_AUTHN_LEVEL_NONE as authentication level before the first DCOM access (even to other DCOM objects than Cndex), otherwise the system automatically calls CoInitializeSecurity with default authentication level, usually CONNECT.

Server Identity: interactive user

This option corresponds to option 1 shown in the previous chapter. In Windows CE systems this option is not relevant. On Windows XP/2000 systems this option must be set from Control Panel->Administrative Tools->Component Services->DCOM Config->cndex->(right mouse click)->Properties->Identity tab.

With this option the remote connection to Cndex can be performed only after a user has logged into the server machine.

Remote Access to Cndex from .NET applications

When a .NET application is run from the Visual Studio IDE, the debug environment automatically calls `CoInitializeSecurity` before starting the application, loading default DCOM security settings. This automation prevents `CndexLink.dll` from setting the DCOM authentication level, because `CoInitializeSecurity` can be invoked only once. Any subsequent call to `CoInitializeSecurity` returns error 0x80010119 (RPC_E_TOO_LATE). For further details:

<http://support.microsoft.com/kb/239561/en-us>

The same thing happens when a .NET application loads a CLR assembly, because some DCOM interface marshalling is required here too.

There are two solutions for this problem that do not require actions on the target machine: the first one is to modify the .NET application so that `CoInitializeSecurity` is called at the proper time, i.e. before any DCOM interface marshalling is performed. The second solution, which is not recommended, involves modifying the DCOM default settings on the machine running the .NET application in order to match the required settings for connection to Cndex. A third solution, which is also the best for developers, requires the user account to be registered on the target machine.

Adapting a .NET application to Cndex

There are two different procedures, one for C# applications and one for VB.NET applications. Both procedures involve adding a source file (`COMInvoke.cs` or `COMInvoke.vb`) to the main application. Choose the source file that matches the application language. Files for both languages can be found in the `UserLibrary` subdirectory of the WinNBI installation directory if WinNBI is installed on the machine. The source file must be added to the main application and not to an assembly referenced by the application itself.

C#

Add `COMInvoke.cs` to your application.

Locate the `Main()` procedure in your application. Usually it is located in `Program.cs`.

At the beginning of the `Main()` function, add the following line:

```
PrimaElectronics.COMUtilities.COMInvoke.InvokeDefaultCoInitializeSecurity();
```

VB.NET

Add `COMInvoke.vb` to your application.

Check if your application is using the application framework events by opening the project properties and selecting the Application tab. If the "Enable application framework" checkbox is checked, your application is using the application framework events.

If your application is using the application framework events:

In the Application tab of the project properties press the View Application Events button.

In the Class Name dropdown box (on the left above the code window) select "My Application Events".

In the Method Name dropdown box (on the right above the code window) select "Startup".

In the `MyApplication_Startup()` function add the following line:

```
PrimaElectronics.COMUtilities.COMInvoke.InvokeDefaultCoInitializeSecurity()
```

If your application is not using the application framework events:

In the main application form (MainFrm in the sample code) add the following code:

```
<STAThread> _
Public Shared Sub Main()
    PrimaElectronics.COMUtilities.COMInvoke. _
        InvokeDefaultCoInitializeSecurity()
    ' Declare a variable named frm of type MainFrm.
    Dim frm As MainFrm
    ' Instantiate (create) a new MainFrm object and assign
    ' it to variable frm.
    frm = New MainFrm ()
    ' Call the Application class' Run method
    ' passing it the MainFrm object created above.
    Application.Run(frm)
End Sub
```

In the Application tab of the project properties window locate the “Startup object” dropdown box and select “Sub Main”.

<End of language specific procedure>

Also make sure that either the Main() function (C#) or Sub Main() subroutine (VB.NET) do not declare any reference variable of a type defined in an assembly (except .NET framework assemblies). This would force the JIT compiler to load the assembly before compiling the Main routine, triggering the DCOM marshalling and basically making the call to InvokeDefaultCoInitializeSecurity() useless. Example:

```
[STAThread]
static void Main()
{
    PrimaElectronics.COMUtilities.COMInvoke.
        InvokeDefaultCoInitializeSecurity();

    Assembly1.A a;
    a = new Assembly1.A();
    Application.Run(new MainFrm(a));
}
```

In the above example, a reference to type A defined in assembly Assembly1 is declared in the Main() function. Since the JIT compiler needs to allocate stack space for all local variables declared in Main() before executing it, Assembly1 is loaded to acquire the declaration of Assembly1.A before the call to InvokeDefaultCoInitializeSecurity() is performed.

To avoid this problem, we need to change the code:

```
[STAThread]
static void Main()
{
    PrimaElectronics.COMUtilities.COMInvoke.
        InvokeDefaultCoInitializeSecurity();

    Application.Run(new MainFrm(new Assembly1.A()));
}
```

The modified Main() function does not declare an explicit reference to Assembly1.A, passing an implicit reference to A directly to the constructor of the main application form instead. This way Assembly1 is not loaded until the constructor of A is called, allowing `InvokeDefaultCoInitializeSecurity()` to initialize DCOM security with the proper settings for Cndex.

Note: this solution does not prevent an application launched from the Visual Studio environment from invoking `CoInitializeSecurity()`, because in this case the call is performed by Visual Studio itself. In order to debug a .NET application that uses Cndex to connect to an OPENControl or PrimaLogic device, the user credentials must be registered on the device, as explained in the next paragraph.

Registering a user with the NTLM protocol on OPENControl and PrimaLogic

This procedure allows the user's credentials to be recognized by the target machine when the application attempts to connect to the Cndex server. Since DCOM uses the NTLM security protocol for authentication purposes, a simple application for registering a user with NTLM is shipped with OPENControl and PrimaLogic machines.

From the OPENControl desktop, press Start->Run;

Type the command:

`\SSD\Osai\XTend\bin\NTLMUser.exe` (OPENControl on WindowsCE 5.0)

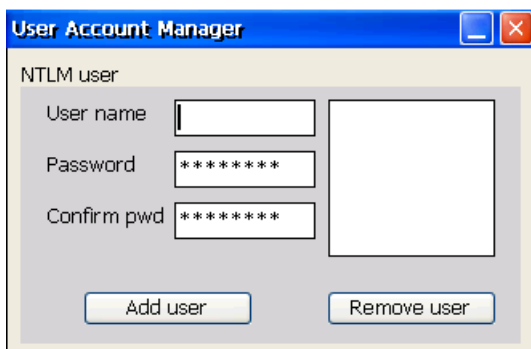
or

`\SSD\Osai\XTend\bin\NTLMUserCE60.exe` (OPENControl on WindowsCE 6.0)

or

`\SSD\Osai\PrimaLogic\bin\NTLMUserCE60.exe` (PrimaLogic)

Press OK. The NTLMUser window will appear.



Type your username and password, then press "Add user". The newly added username will appear in the right box of the NTLMUser window.

Press Start->Programs->Regflush to make the user registration permanent.

NOTE: the account password is saved in the registry in 3DES encoding format so that it cannot be easily stolen by an unauthorized user gaining access to the target machine.

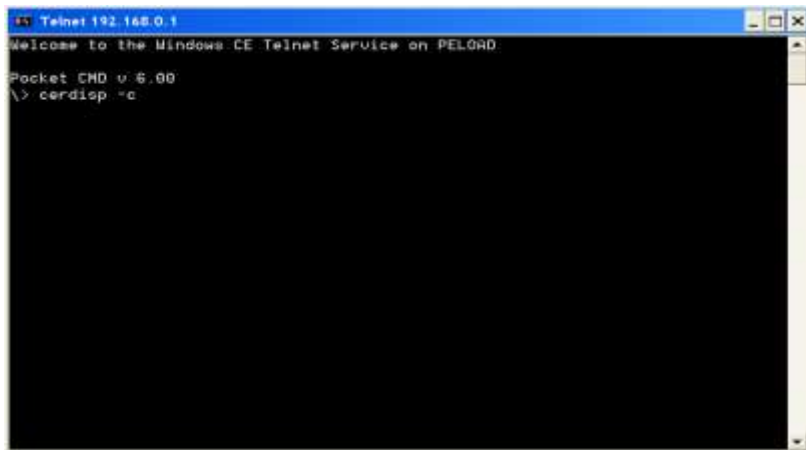
Remote Desktop on PrimaLogic

Unlike OPENControl, display and keyboard cannot be plugged into PrimaLogic. However, the Windows CE desktop on PrimaLogic can still be accessed from a PC connected to the target through LAN network (better if point-to-point connection with a crossover cable). On the PC open a command prompt and create a telnet connection to the PrimaLogic by typing:

`telnet <IP address of the PrimaLogic>` (usually 192.168.0.1)

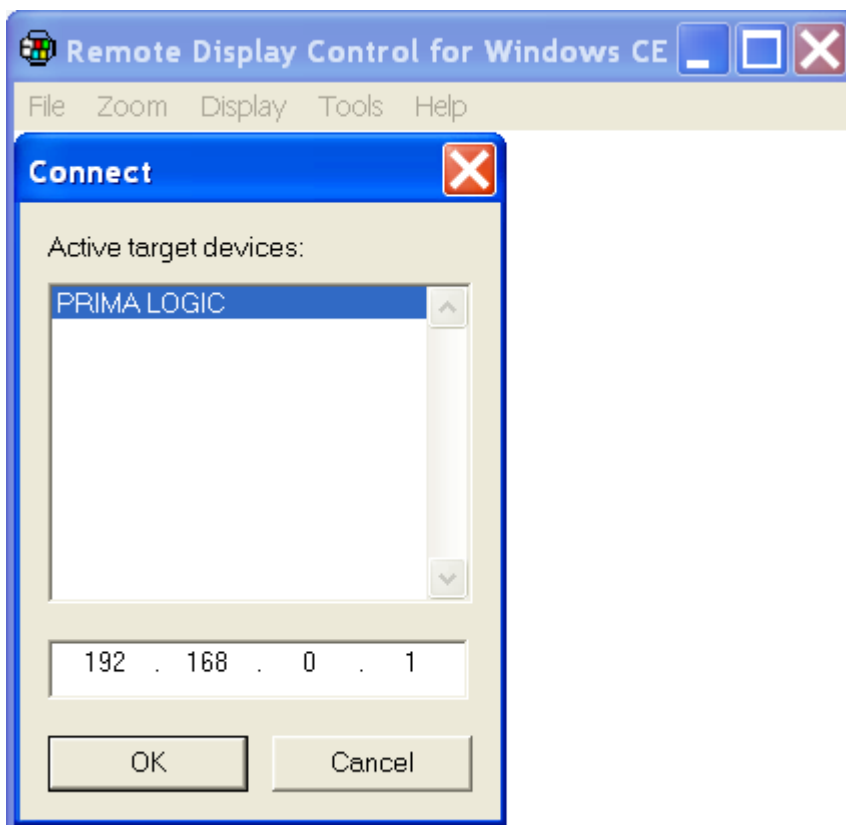
Upon successful telnet connection type:

`cerdisp -c`



NOTE: do not close the command prompt window until you are done with Remote Desktop.

Now find the cerhost_CE60.exe file in the Utility subdirectory of the WinNBI installation directory and launch it. From the File menu select Connect. After a few seconds "PRIMA LOGIC" should appear in the upper box. Select it then press OK and Remote Desktop is ready for use. The procedure for user registration from now is the same as for OPENControl.



Registering a user with the NTLM protocol on CeWin

This procedure is almost identical to the one described in the previous paragraph for OPENControl. However, it is necessary to add data to the WindowsCE system registry by editing the CeWin configuration files. The procedure must be executed on the CeWin host system, i.e. the system where CeWin is actually running rather than the system where the application is running.

With a text editor, open the Os.config file located in the CeWin installation directory.

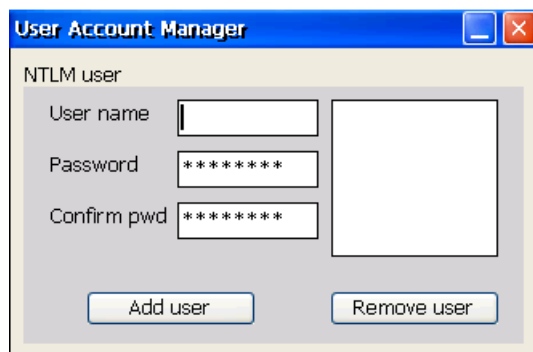
1. Append the following text to the file:

```
[HKEY_LOCAL_MACHINE\init\BootVars]
"MasterKeyFileDir"="\Network\SSD\Osai\\"
```

2. Shutdown and restart CeWin.
3. From the CeWin remote desktop, press Start->Run.
4. Type the command:

```
\Network\SSD\Osai\XTend\bin\NTLMUserCE60.exe
```

Press OK. The NTLMUser window will appear.



5. Type your username and password, then press "Add user". The newly added username will appear in the right box of the NTLMUser window.
6. From the CeWin remote desktop, press Start->Run.
7. Type the command:

```
\Network\SSD\Osai\XTend\bin\ReadNTLM.exe
```

8. From the CeWin host system, open the file named ntlmkey.txt, located in the \Network\SSD folder. Copy the two lines corresponding to the newly added user:

```
[HKEY_LOCAL_MACHINE\Comm\Security\UserAccounts\<username>]
"NT"=hex:01,00.....
```

9. Append the two lines to the Os.config file.

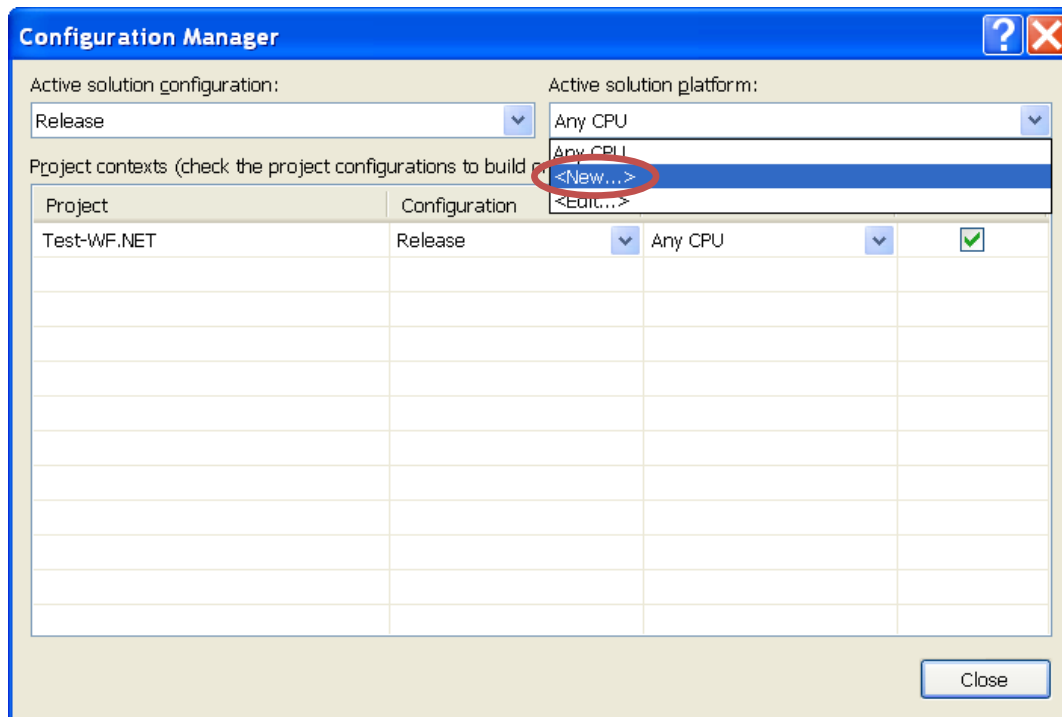
NOTE: the account password is saved in the registry in 3DES encoding format so that it cannot be easily stolen by an unauthorized user gaining access to the target machine.

Compatibility of .NET applications with Cndex on 64 bit operating systems

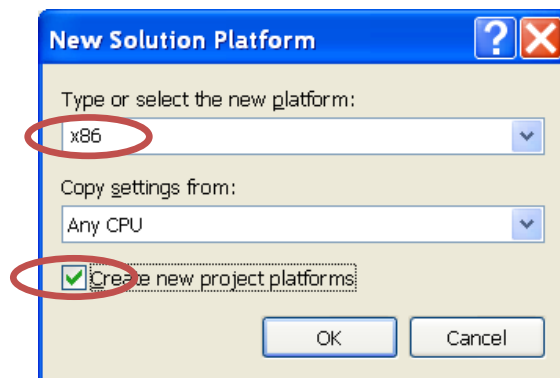
Both WinNBI and Cndex are distributed in their 32 bit version only. Although 64 bit operating systems like Windows 7 64 bit are fully capable of running 32 bit code, there are in fact some limitations. In particular, a 64 bit executable cannot dynamically link a 32 bit library, and vice versa.

If a .NET executable is built using the default “Any CPU” platform, the JIT compiler automatically adapts it to the type of operating system in order to get the best performance. In short, if the executable runs on a 64 bit system the code is automatically compiled for 64 bit by the JIT compiler. Since Cndex and its related DLLs are only available in the 32 bit version, a 64 bit application trying to use Cndex generates a “DLL not found” exception, even if WinNBI is correctly installed on the machine. To overcome this problem the Visual Studio project must be configured to build an executable that runs only in 32 bit mode, regardless of the operating system architecture.

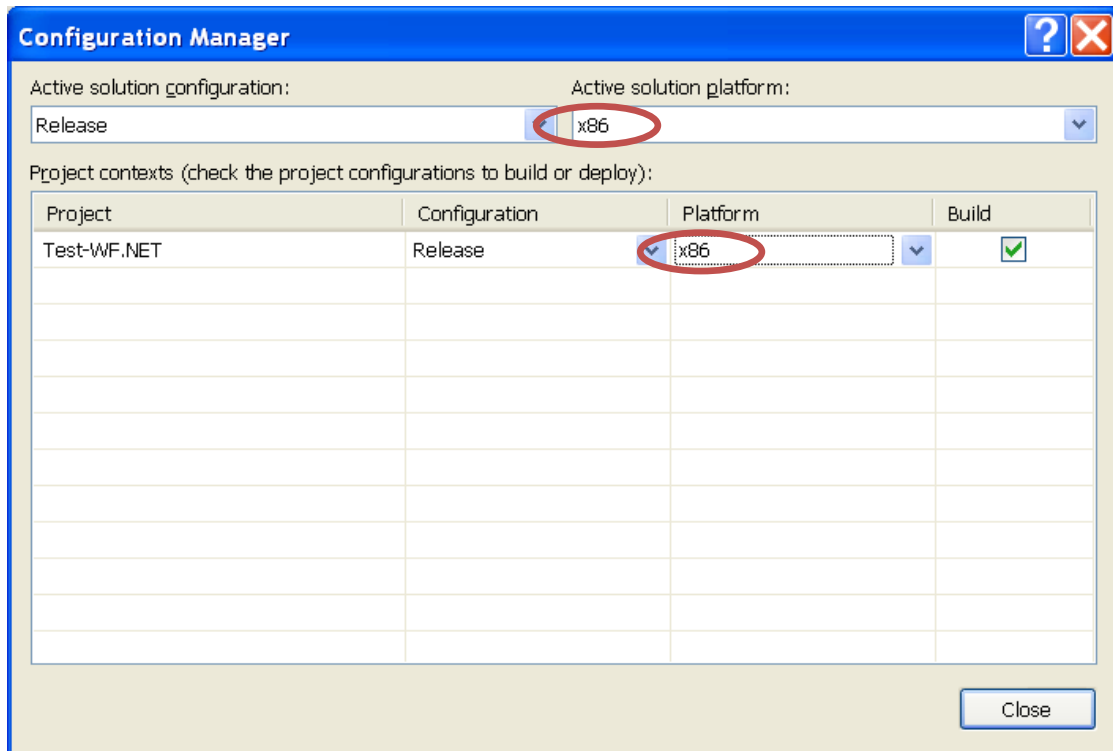
1. Open the .NET project in Visual Studio, then select Build->Configuration Manager from the menu.
2. In the Configuration Manager window, from the Active Platform select <New...>.



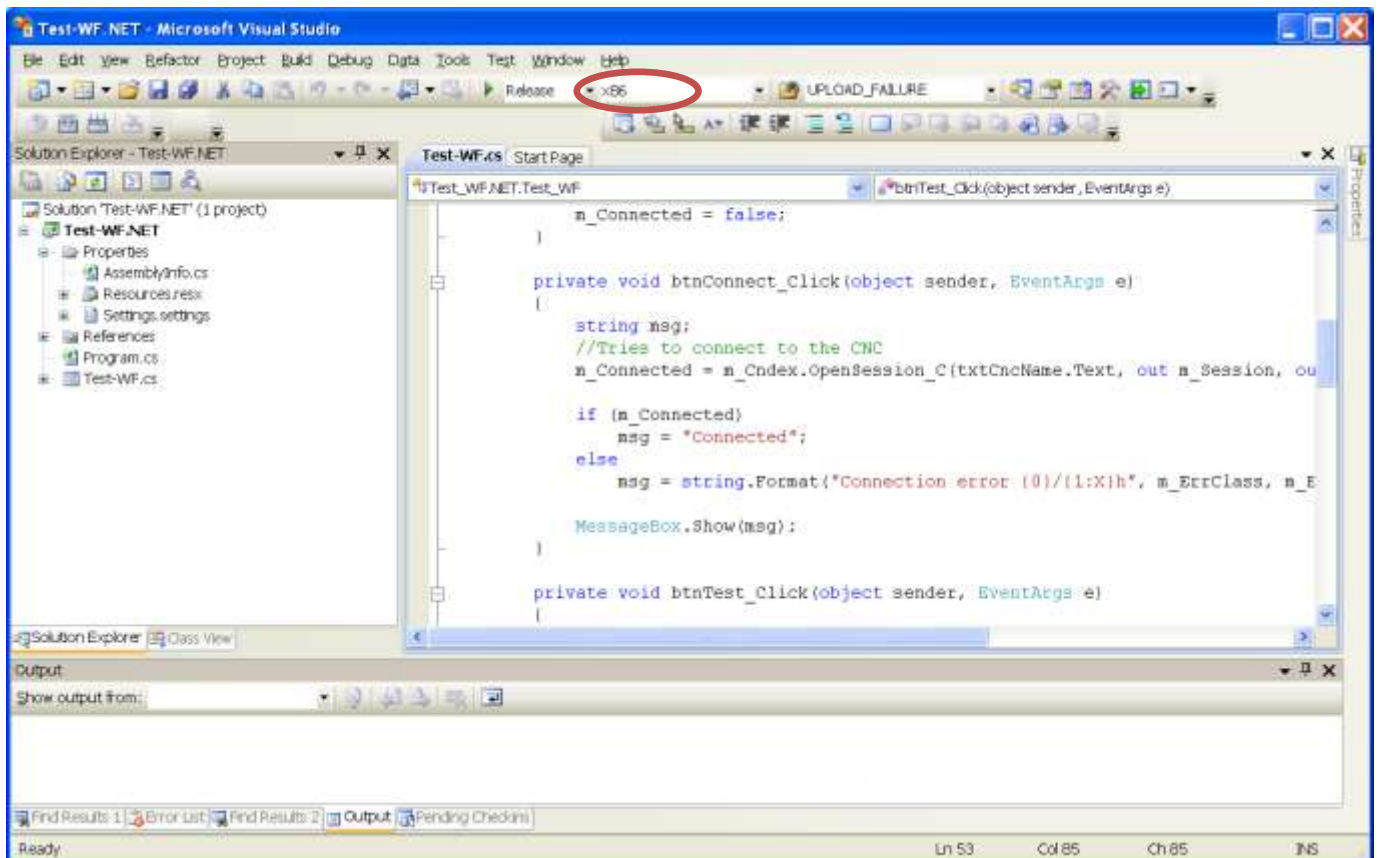
3. From the dialog box select “x86” as new platform. Make sure that the “Create new project platforms” flag is checked, then press OK.



4. Check that the project platform “x86” matches the solution platform “x86” for both the Debug and Release configurations.



5. In the main window of Visual Studio select “x86” as active solution platform and build the project.

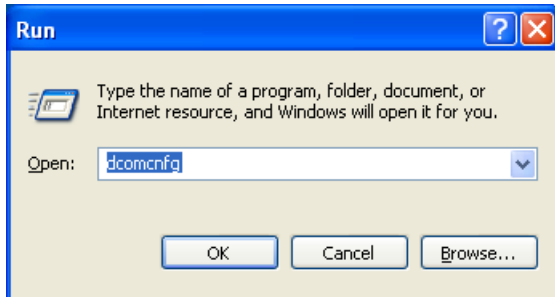


Enabling remote access to Cndex on a desktop system

The content in this section only applies to desktop systems that need to be used as Cndex servers, for example when a remote machine needs to access the logical file system of the local host, or the local host must work as a Cndex proxy for connecting to a Series10 system from a remote machine.

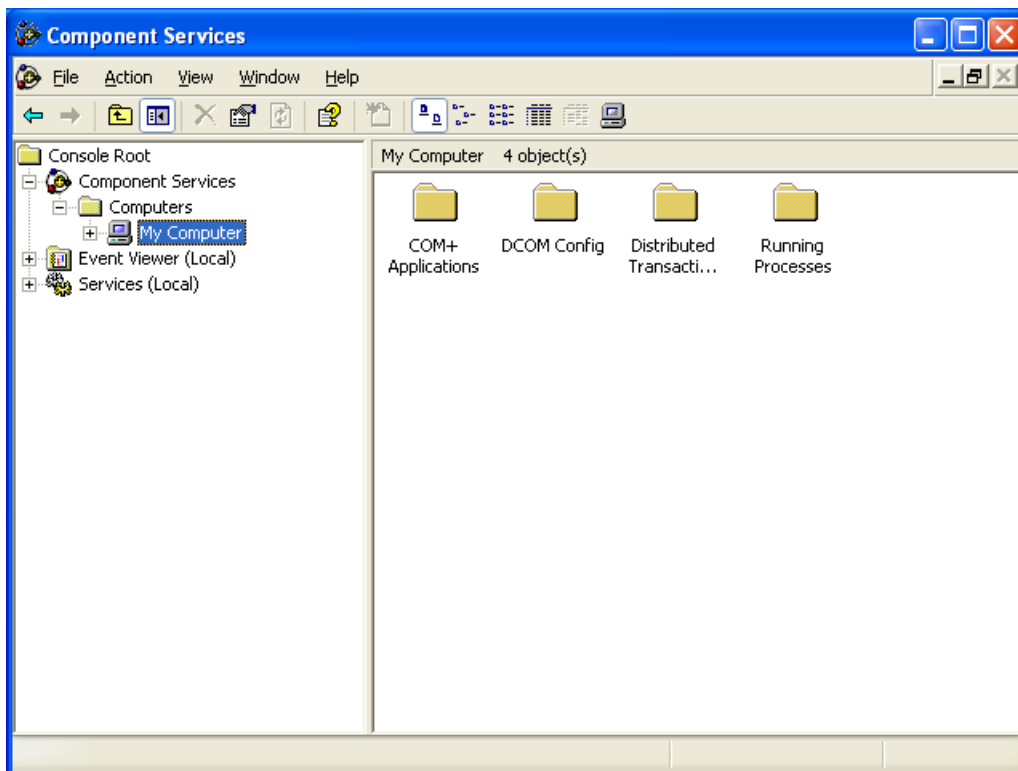
Configuring DCOM on the server system

From Control Panel open Administrative Tools->Component Services. Alternatively run “dcomcnfg” from the command prompt.



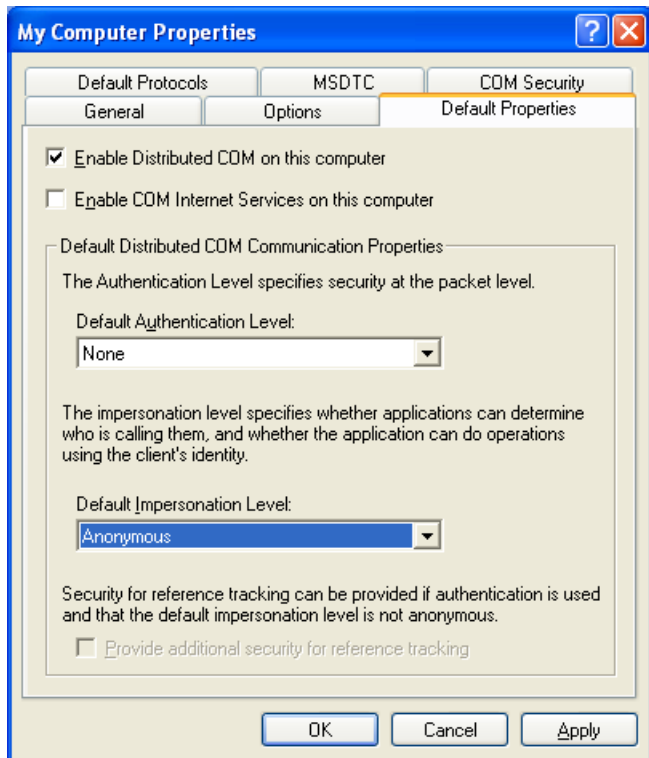
Enabling DCOM

From “Component Services” select Computer->My Computer. By right-clicking My Computer select Properties.

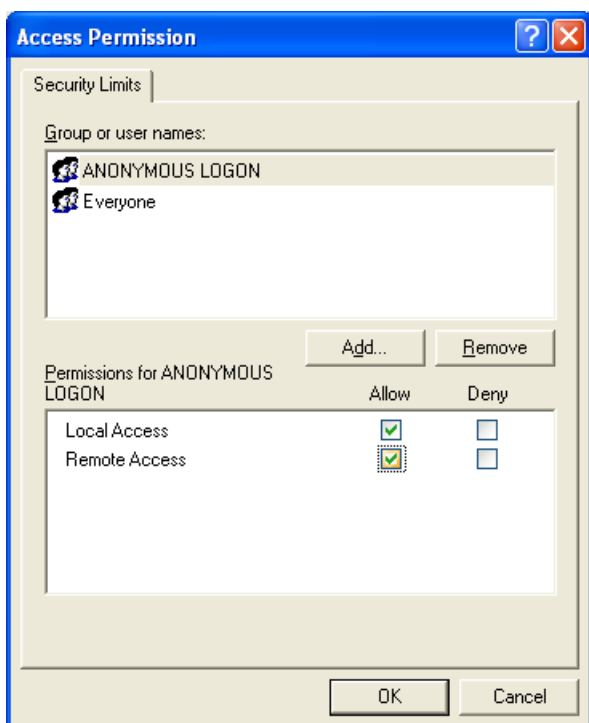
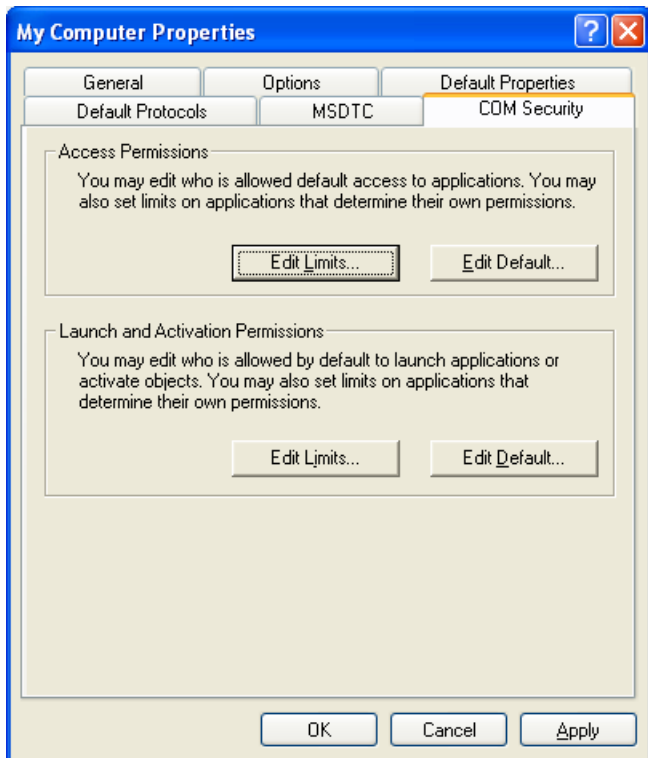


From the Properties panel select the “Default Properties” tab:

- ❑ check the “Enable Distributed COM on this computer” checkbox;



Select the “COM Security” tab. In the “Access Permissions” frame press the “Edit Limits” button. From the Access Permissions panel insert the ANONYMOUS LOGON user, if not already present. Enable all access rights for this user and press OK.

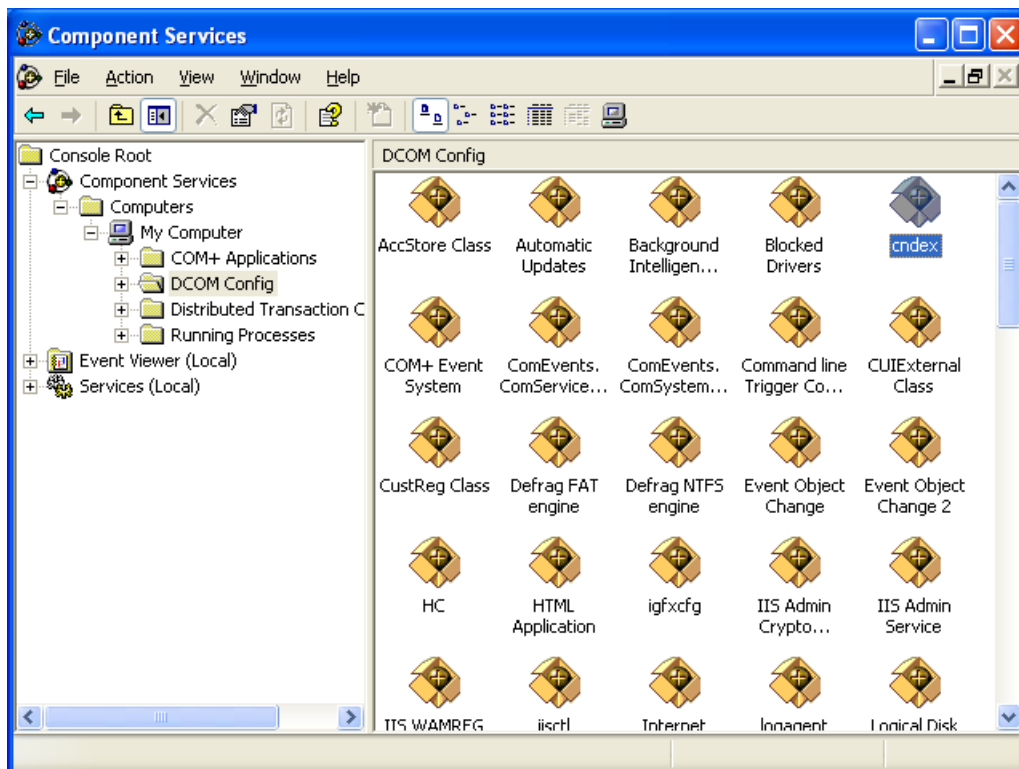


In the “Launch and Activation Permissions” frame press the “Edit Limits” button and repeat the same operation. Once done press OK in the Properties panel of My Computer.

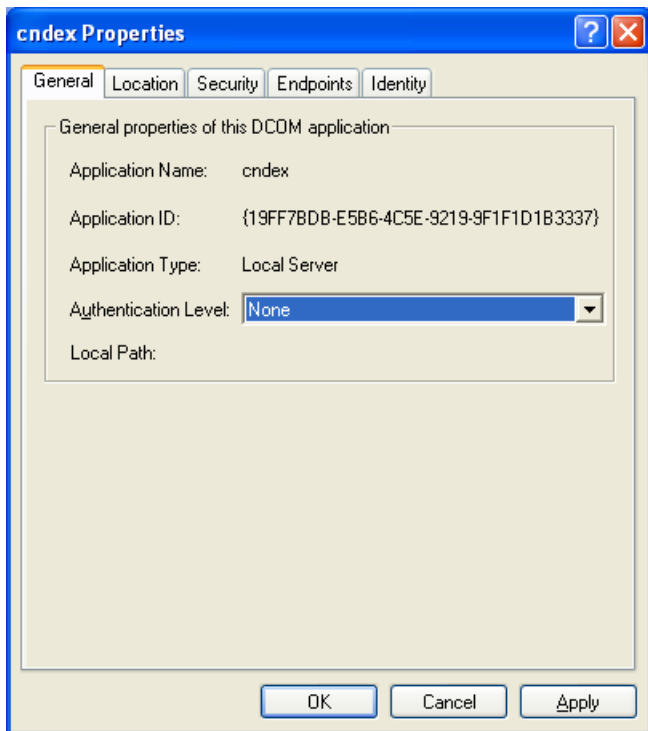
The configuration of DCOM default security settings is done. Now the Cndex specific security settings must be configured.

Cndex configuration

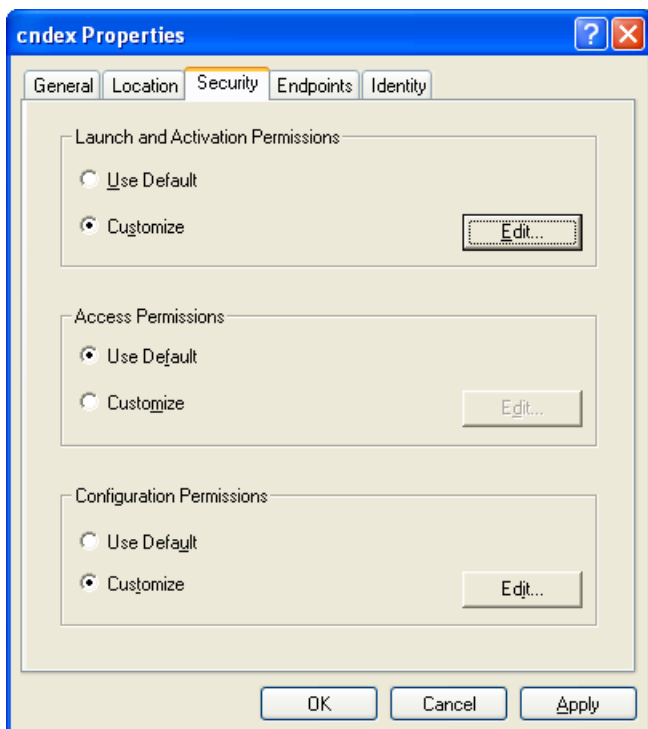
From Component Services select DCOM Config. In the right panel select “cndex”. Right click and open the Properties panel of cndex.



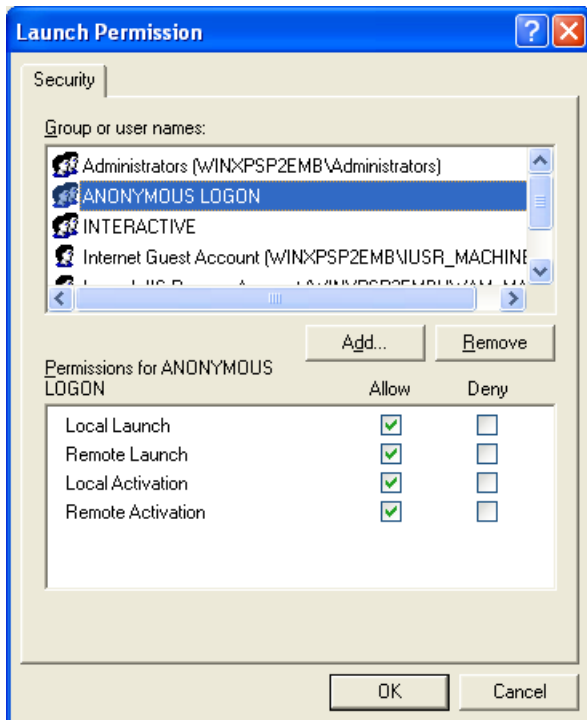
From the Properties panel of cndex select the General tab. Select Authentication Level “None”.



Select the “Security” tab. in the “Launch and Activation Permissions” frame select “Customize” then press the “Edit” button.



From the “Launch Permissions” panel add the ANONYMOUS LOGON user, if not already present, and enable all rights for it. Then add the SYSTEM user, if not already present, and enable all rights for it. Press OK when done.



Repeat the operation in the “Access Permissions” and “Configuration Permissions” frames of the Properties panel of cndex.

Select the “Identity” tab. Select the radio button “The interactive user”. Press OK in the Properties panel of cndex. Cndex configuration is done.

NOTE: with the Identity settings selected it is necessary that a user is logged into the system in order to use it as a Cndex server.

