

ADONTEC's Interprocess Communication Library (ADIPC)

Test Run

ADIPC offers a easy to use and portable API for fast and rock solid inter process communications.

The learnig curve is low. You can extend your software with ipc capabilities in very short time.

This document describes an IPC test run using different clients compiled for different CPU modes (32, 64 bit).

The operating system used was Windows 10 Pro, x64. Windows 64 bit was selected for this run in order to run 32 and 64 bit software the same time.

The IPC applications were 32 and 64 bit written in C/C++, C#, Delphi and VB net.

Since the ADIPC library is portable one can use the same source code and only need to change compiler switches. Also the 32 bit ADIPC library does execute under x64 without limitations.

Any application using the ADIPC Library is able to request and reply data packages. Thus it is by design a server and client application. We can limit one application to run as a server only by just not sending requests. In the following examples the application using the endpoint name "Pier02" is acting as a server only.

```
D:\Working\ADIPC\VC6\x64\tstipc_64.exe
Received [REQ=What time is it?]
Received [REQ=What time is it ? from Pier51]
Received [REQ=What time is it?]
Received [REQ=What time is it ? from Pier51]
Received [REQ=What time is it?]
Received [REQ=What time is it?]
Received [REQ=What time is it?]
```

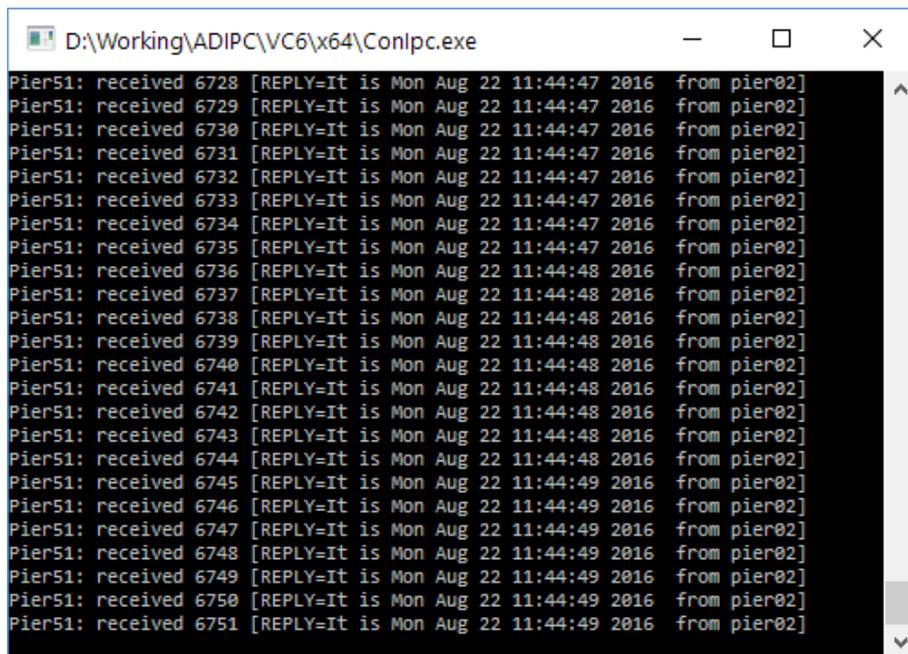
A C/C++ written 64 bit IPC Server named „Pier02“.

The clients are sending the same request as it is the one this server knows to serve: „REQ=What Time is it ?“

In this run (as we can see) the client „Pier51“ also adds his name at the end of the request line. It is informational only.

The request and reply strings can be any data of course the applications know and understand how to handle This is purely custom functionality.

The tstipc project compiles for 32 bit and 64 bit using the same source code.

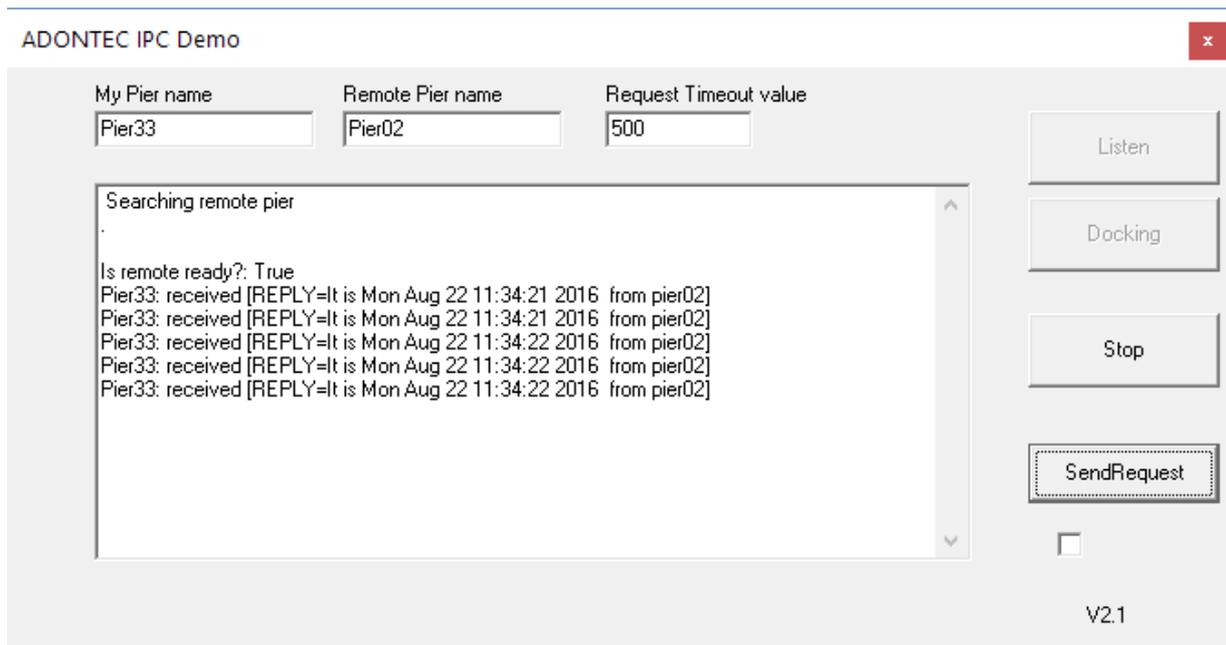


```
D:\Working\ADIPC\VC6\x64\Conlpc.exe
Pier51: received 6728 [REPLY=It is Mon Aug 22 11:44:47 2016 from pier02]
Pier51: received 6729 [REPLY=It is Mon Aug 22 11:44:47 2016 from pier02]
Pier51: received 6730 [REPLY=It is Mon Aug 22 11:44:47 2016 from pier02]
Pier51: received 6731 [REPLY=It is Mon Aug 22 11:44:47 2016 from pier02]
Pier51: received 6732 [REPLY=It is Mon Aug 22 11:44:47 2016 from pier02]
Pier51: received 6733 [REPLY=It is Mon Aug 22 11:44:47 2016 from pier02]
Pier51: received 6734 [REPLY=It is Mon Aug 22 11:44:47 2016 from pier02]
Pier51: received 6735 [REPLY=It is Mon Aug 22 11:44:47 2016 from pier02]
Pier51: received 6736 [REPLY=It is Mon Aug 22 11:44:48 2016 from pier02]
Pier51: received 6737 [REPLY=It is Mon Aug 22 11:44:48 2016 from pier02]
Pier51: received 6738 [REPLY=It is Mon Aug 22 11:44:48 2016 from pier02]
Pier51: received 6739 [REPLY=It is Mon Aug 22 11:44:48 2016 from pier02]
Pier51: received 6740 [REPLY=It is Mon Aug 22 11:44:48 2016 from pier02]
Pier51: received 6741 [REPLY=It is Mon Aug 22 11:44:48 2016 from pier02]
Pier51: received 6742 [REPLY=It is Mon Aug 22 11:44:48 2016 from pier02]
Pier51: received 6743 [REPLY=It is Mon Aug 22 11:44:48 2016 from pier02]
Pier51: received 6744 [REPLY=It is Mon Aug 22 11:44:48 2016 from pier02]
Pier51: received 6745 [REPLY=It is Mon Aug 22 11:44:49 2016 from pier02]
Pier51: received 6746 [REPLY=It is Mon Aug 22 11:44:49 2016 from pier02]
Pier51: received 6747 [REPLY=It is Mon Aug 22 11:44:49 2016 from pier02]
Pier51: received 6748 [REPLY=It is Mon Aug 22 11:44:49 2016 from pier02]
Pier51: received 6749 [REPLY=It is Mon Aug 22 11:44:49 2016 from pier02]
Pier51: received 6750 [REPLY=It is Mon Aug 22 11:44:49 2016 from pier02]
Pier51: received 6751 [REPLY=It is Mon Aug 22 11:44:49 2016 from pier02]
```

A C# written 64 bit IPC client named „Pier51“ connected to the x64 IPC Server „Pier02“.

It sends out a request „REQ=What Time is it ? from Pier51“ and receives the reply „REPLY=It is .. from Pier02“.

Pier02 is replying with its name at the end of the line.



A Delphi written 32 bit GUI application named „Pier33“ connecting to the 64 Bit IPC Server named „Pier02“.

It sends out a request „REQ=What Time is it ?“ and expects the reply as shown above.

By pressing the **Listen** button the application starts listening for requests at “Pier33”. In order to submit requests we “connect” to the remote station named “Pier02” by pressing the **Docking** button. This is more or less the ADIPC_QueryRemoteAddress executing and providing the remote station (“Pier02”) address so this application can transmit requests also.

```
D:\Working\ADIPC\VC6\x64\tstipc_64.exe
Received reply [REPLY=It is Mon Aug 22 12:32:13 2016 from pier02] t=0
Received reply [REPLY=It is Mon Aug 22 12:32:13 2016 from pier02] t=0
Received reply [REPLY=It is Mon Aug 22 12:32:13 2016 from pier02] t=0
Received reply [REPLY=It is Mon Aug 22 12:32:13 2016 from pier02] t=0
Received reply [REPLY=It is Mon Aug 22 12:32:13 2016 from pier02] t=0
Received reply [REPLY=It is Mon Aug 22 12:32:13 2016 from pier02] t=0
Received reply [REPLY=It is Mon Aug 22 12:32:13 2016 from pier02] t=0
Received reply [REPLY=It is Mon Aug 22 12:32:13 2016 from pier02] t=0
Received reply [REPLY=It is Mon Aug 22 12:32:13 2016 from pier02] t=1
Received reply [REPLY=It is Mon Aug 22 12:32:13 2016 from pier02] t=0
Received reply [REPLY=It is Mon Aug 22 12:32:13 2016 from pier02] t=0
Received reply [REPLY=It is Mon Aug 22 12:32:13 2016 from pier02] t=0
Received reply [REPLY=It is Mon Aug 22 12:32:13 2016 from pier02] t=0
Received reply [REPLY=It is Mon Aug 22 12:32:13 2016 from pier02] t=0
Received reply [REPLY=It is Mon Aug 22 12:32:13 2016 from pier02] t=1
Received reply [REPLY=It is Mon Aug 22 12:32:13 2016 from pier02] t=0
Received reply [REPLY=It is Mon Aug 22 12:32:13 2016 from pier02] t=0
Received reply [REPLY=It is Mon Aug 22 12:32:13 2016 from pier02] t=0
Received reply [REPLY=It is Mon Aug 22 12:32:13 2016 from pier02] t=0
Received reply [REPLY=It is Mon Aug 22 12:32:13 2016 from pier02] t=0
Received reply [REPLY=It is Mon Aug 22 12:32:13 2016 from pier02] t=1
Received reply [REPLY=It is Mon Aug 22 12:32:13 2016 from pier02] t=0
Received reply [REPLY=It is Mon Aug 22 12:32:13 2016 from pier02] t=0
Received reply [REPLY=It is Mon Aug 22 12:32:13 2016 from pier02] t=0
```

In this run additional 64 bit clients are connected to „Pier02“ named „Pier11“ and „Pier33“.

As one can easily observe the example `tstipc_64.exe` supports client and server mode. All IPC application using the ADONTEC's Interprocess Library can do this automatically.

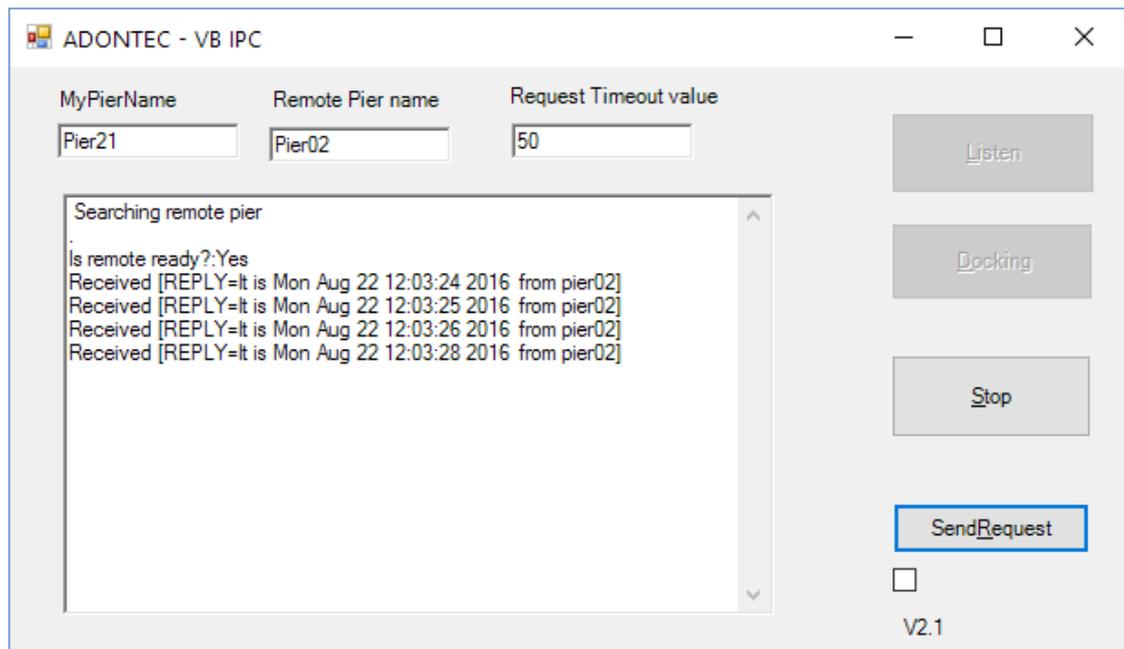
The IPC application „Pier02“ is running as a pure server only because it does not submit requests if the name is „Pier02“. So, in this sample, a simple „If (bServerOnly)“ is what is suppressing it based on the selected name for the endpoint (“Pier02”).

Timing

The “t=0” or “t=1” we see at the end of each line is the time in ms taken from submitting the request to receive and handle the reply.

This time stamp looks pretty low and it is very low for nearly all the requests we did here having about five clients fetching from the same server.

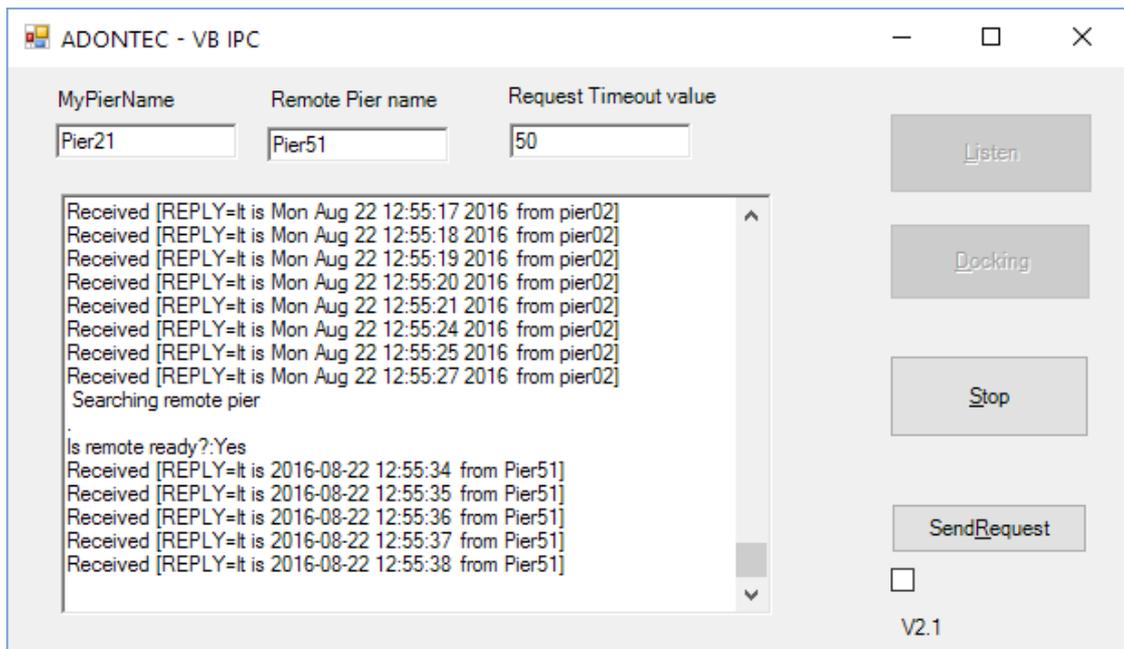
But, one must remember that Windows is no real time operating system and it may occur that one request may take longer e. g. +10ms based on the Windows sheduler's granularity and it can also get worse if the system is stressed to much or on a slow CPU.



A VB net written 32 bit GUI application. The same project also compiles to 64 Bit by just switching compiler settings and using the same source code.

By pressing the **Listen** button the application starts listening for requests at "Pier21".

In order to submit requests we "connect" to the remote station named "Pier02" by pressing the **Docking** button. This is more or less the ADIPC_QueryRemoteAddress executing and providing the remote station ("Pier02") address so this application can transmit requests also.



The same application connected to "Pier51" instead.

Summary

We've tested compatibility and speed between 32 and 64 bit IPC applications.

Every connection is defined by two endpoints (local application and remote application). The endpoint "Pier02" was used as a pure server.

Note

Please remember that there is no such limitation that all clients have to connect to the same endpoint "pier02" as we did in this test run. Any application can connect to any available endpoint since all endpoints are always listening. So we could easily change the remote name to "Pier51" e. g. in the VB net sample named "Pier21" and connect to "Pier51" instead of "Pier02".

Once the IPC application is started it is in listening state. In order to transmit requests it must "connect" to one or more remote stations using ADIPC_QueryRemoteAddress.

Endpoint

Is the name an application, using ADIPC, is listening to ("service name").

The endpoint name must be unique in the system. ADIPC_StartListen will not allow the same endpoint name appearing twice.

No limits

There is no limitation on the number of connections the application is listening to or it "connects" to.

Portable API

Once you have a running application you can compile the same source code to 32 or 64 bit. The ADIPC API is portable on source level.

In order to execute the application you also have to use the correct ADIPC.DLL (32/64 bit).

Runtime

The C/C++, C# and Delphi samples are using the adipc.dll. The VB net is also using the ADIPCClassLibrary.dll, which includes the classes IPC and TADIPC.

Both dlls (adipc.dll , ADIPCClassLibrary.dll) are the so called runtime files and can shipped with the customers IPC application.