Open NFC - Connection Center - User's Manual

Document Type:	Manual
Reference:	MAN_NFC_0904-106 Version 1.4 (14516)
Release Date:	Jan. 27, 2012
File Name:	MAN_NFC_0904-106 Open NFC - Connection Center - User's Manual.pdf
Security Level:	General Business Use

Disclaimer

History

Version	Date	Comments
0.1	April 1, 2009	First Draft
1.0	April 2, 2009	Release for Open NFC 3.1
1.1	Feb. 1, 2010	Add information about the RS232 Connector. Add information about the Trace server auto start function.
1.2	Dec. 8, 2010	Release for Open NFC 4.1
1.3	May 20, 2011	New document template
1.4	Jan. 27, 2012	Review document to new release of Connection Center

Summary of Contents

1 Introduction

The connection center is a tool simplifying the connection of the numerous services in the Open NFC tool kit. A connection center instance is executed in every physical PC and in every virtual PC used for the Open NFC toolkit.

Each executable providing a service connects to the local connection center hosting the executable to declare the service. Each executable needing a service connects to the connection center hosting the executable to requests the service. When a client requests a service, the connection center finds the connection to the locally hosted service provider or connects to another connection center to look for remote service providers.

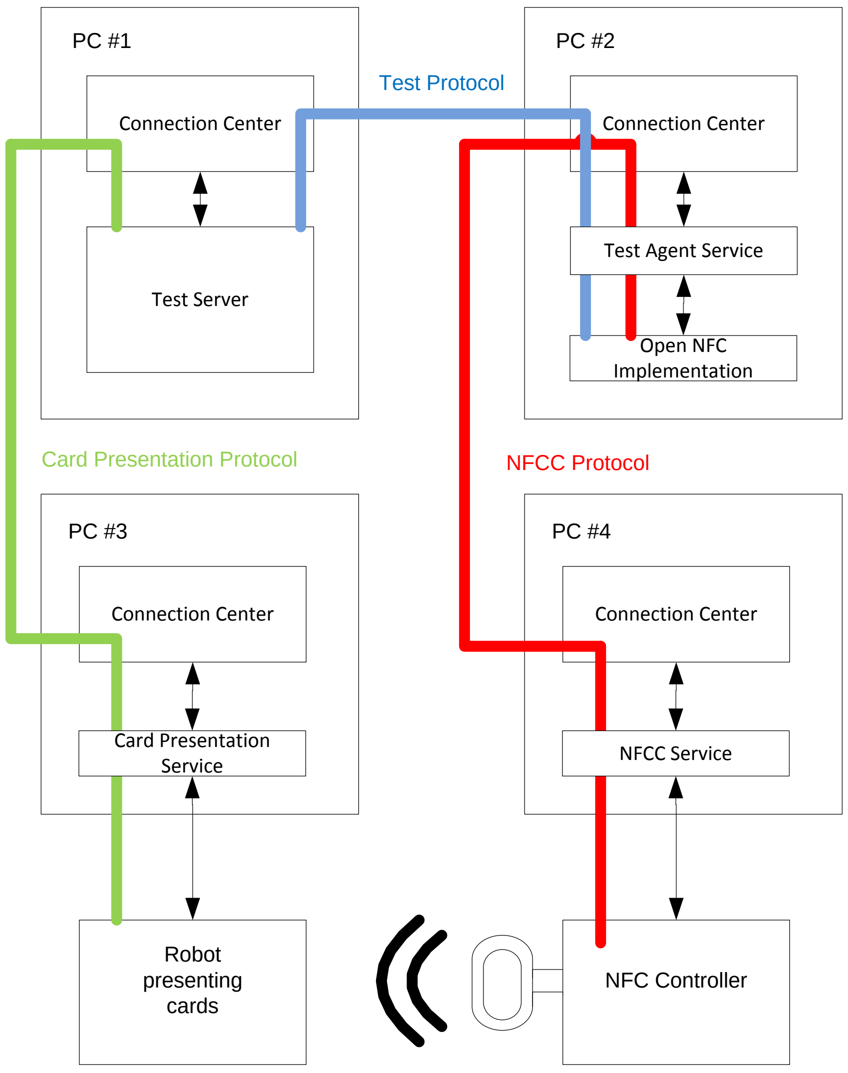


Figure 1 – Example of Configuration

The following services are defined:

- Trace Service
- SHDLC Protocol
- NFC HAL Protocol
- Test Protocol
- Card Presentation Service
- ISO7816-4 Card Service
- ISO14443-4 A Card Service
- ISO14443-4 B Card Service
- ISO14443-3 A Card Service
- ISO14443-3 B Card Service
- P2P Target Service
- Type 1 Card Service
- Type 3 Card Service
- ISO15693-3 Card Service

2 Protocol description

The protocol with the connection center relies on TCP-IP. There are three types of executables connecting to a connection center:

- a service provider,
- a service client (service requester), or
- another connection center in another PC or virtual PC

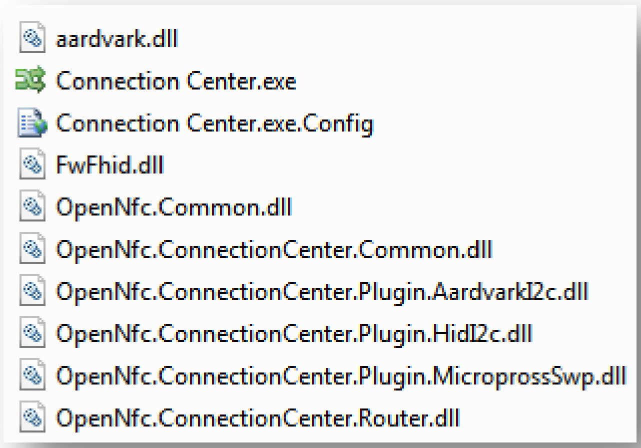
3 Connection Center

3.1 Installation and launching

The Connection Center “***Connection Center.exe”*** can be installed in any folder.

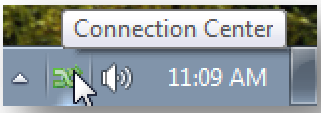
From our delivery package, you can find a zip file like :
“Open NFC - Core Edition v4.4.0 (12682).zip”

Extract it and go to the “\core\connection_center\” folder to find the Connection Center. There is no installation procedure; just ensure that the following files are in the destination directory:

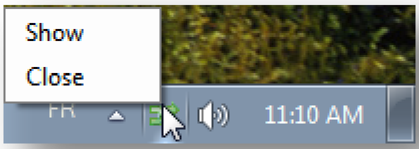


aardvark.dll
Connection Center.exe
Connection Center.exe.Config
FwFhid.dll
OpenNfc.Common.dll
OpenNfc.ConnectionCenter.Common.dll
OpenNfc.ConnectionCenter.Plugin.AardvarkI2c.dll
OpenNfc.ConnectionCenter.Plugin.HidI2c.dll
OpenNfc.ConnectionCenter.Plugin.MicroprossSwp.dll
OpenNfc.ConnectionCenter.Router.dll

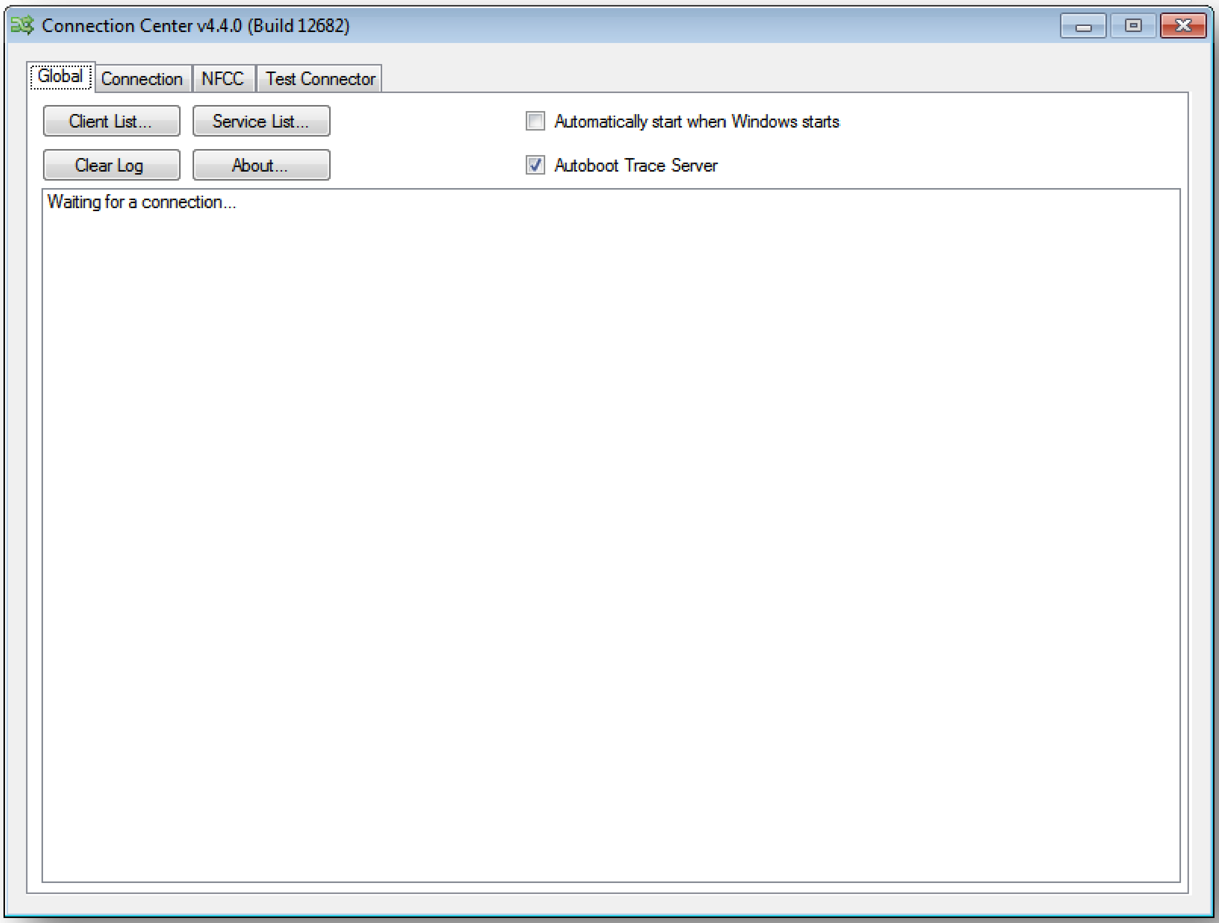
After launching the executable, only an icon in the notification area Windows appears:



Right click on it and you see a menu:



Click on “Close” to shut down the Connection Center.
Click On “Show” to display the main console management of the Connection Center:



☐ Automatically start when Windows starts
☑ Autoboot Trace Server

```
Waiting for a connection...
```

By clicking the “Minimize” button in the upper right of the window, the Connection Center is hiding.

The main windows give access to four tabs:

- [Global] : displays activity log, and allows to monitor the current Service Providers and Service Clients that are connected,
- [Connection] : customizing the way the Connection Center accepts connections from remote machines, and if the Connection Center uses a parent Connection Center,
- [NFCC]: the main screen for connecting to a predefined local NFC Controller.
- [Test Connector] : connecting to a custom test connector

3.2 Using the [Global] Tab

The "Global" tab displays activity log (connection/disconnection events to the Connection Center, errors …) and gives a way to show the list of service providers and service clients that are currently connected to the connection center.

The following functionalities are accessible through this "Global" tab:

a. Automatically start when Windows starts.
   If you check this box, the Connection Center will be started every time you start your machine

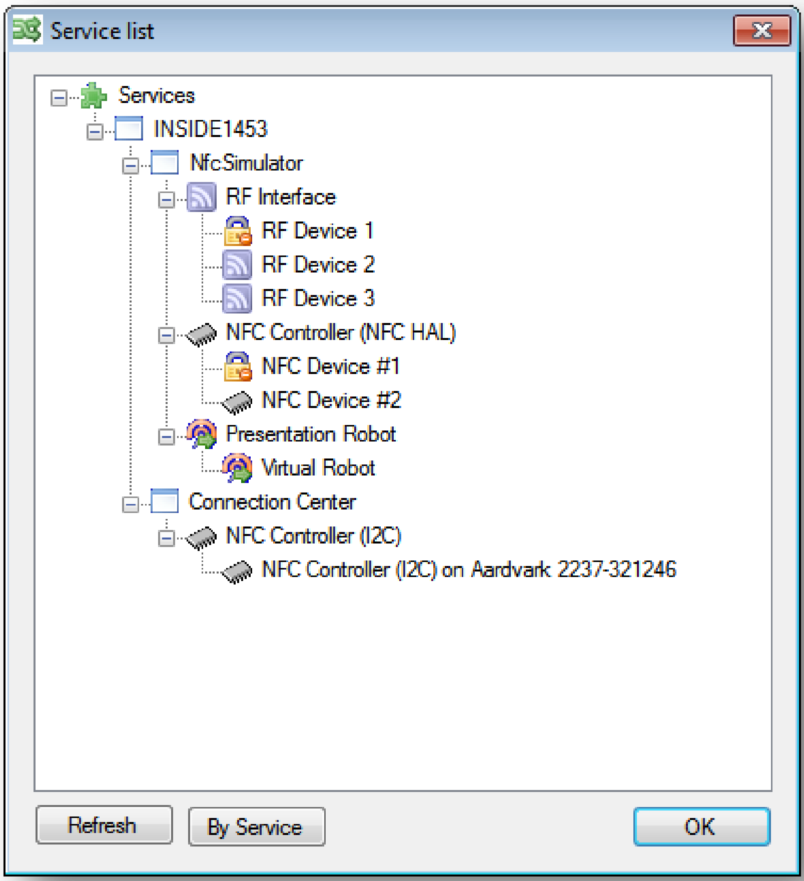
b. Auto boot trace server.
   Start the tracer server application when application needs displaying a trace.

c. Clear log
   To clear the Log area, push the [Clear Log] button.

d. Service List and Client List

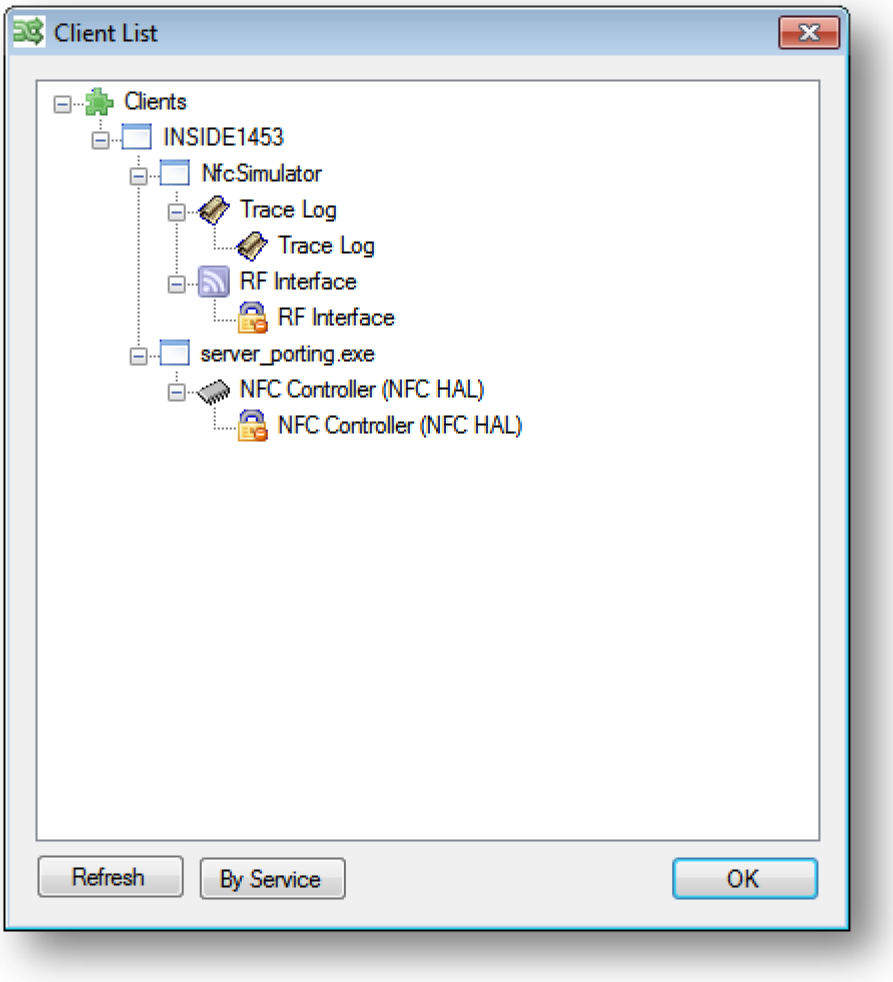
To have the exhaustive list of Service Providers and Service Clients connected to the Connection Center:

- Press the [Service List …] button.



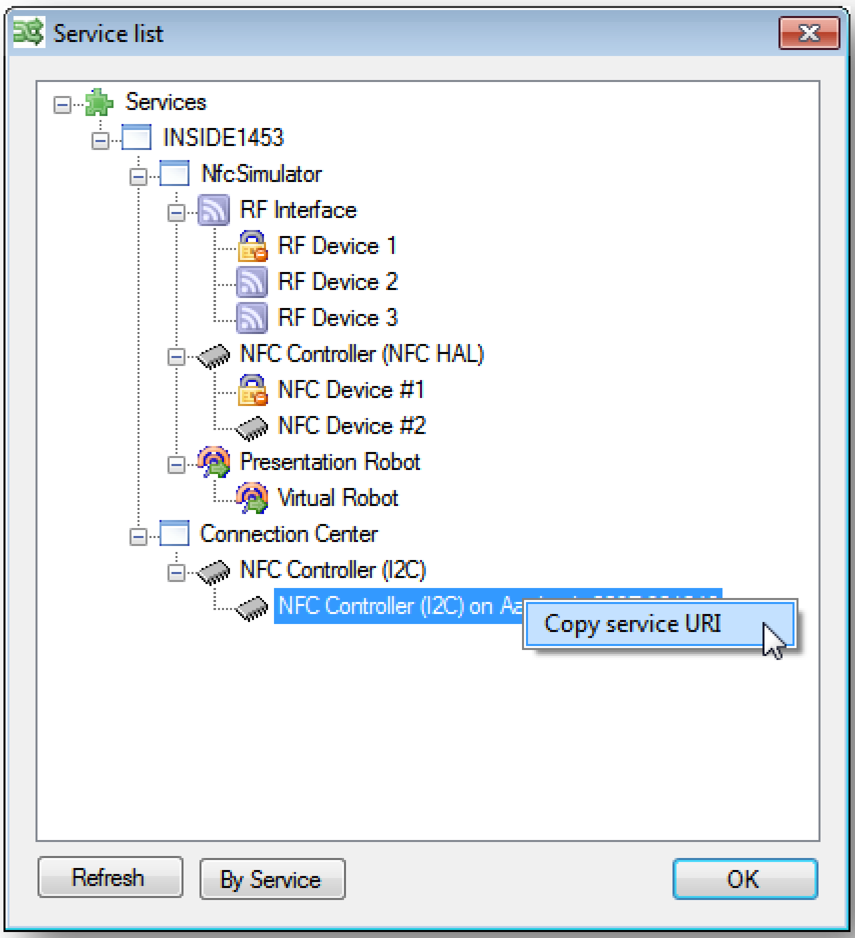
```
Services
  INSIDE1453
    NfcSimulator
      RF Interface
        RF Device 1
        RF Device 2
        RF Device 3
      NFC Controller (NFC HAL)
        NFC Device #1
        NFC Device #2
      Presentation Robot
        Virtual Robot
    Connection Center
      NFC Controller (I2C)
        NFC Controller (I2C) on Aardvark 2237-321246
```

- Press the [Client List …] button.



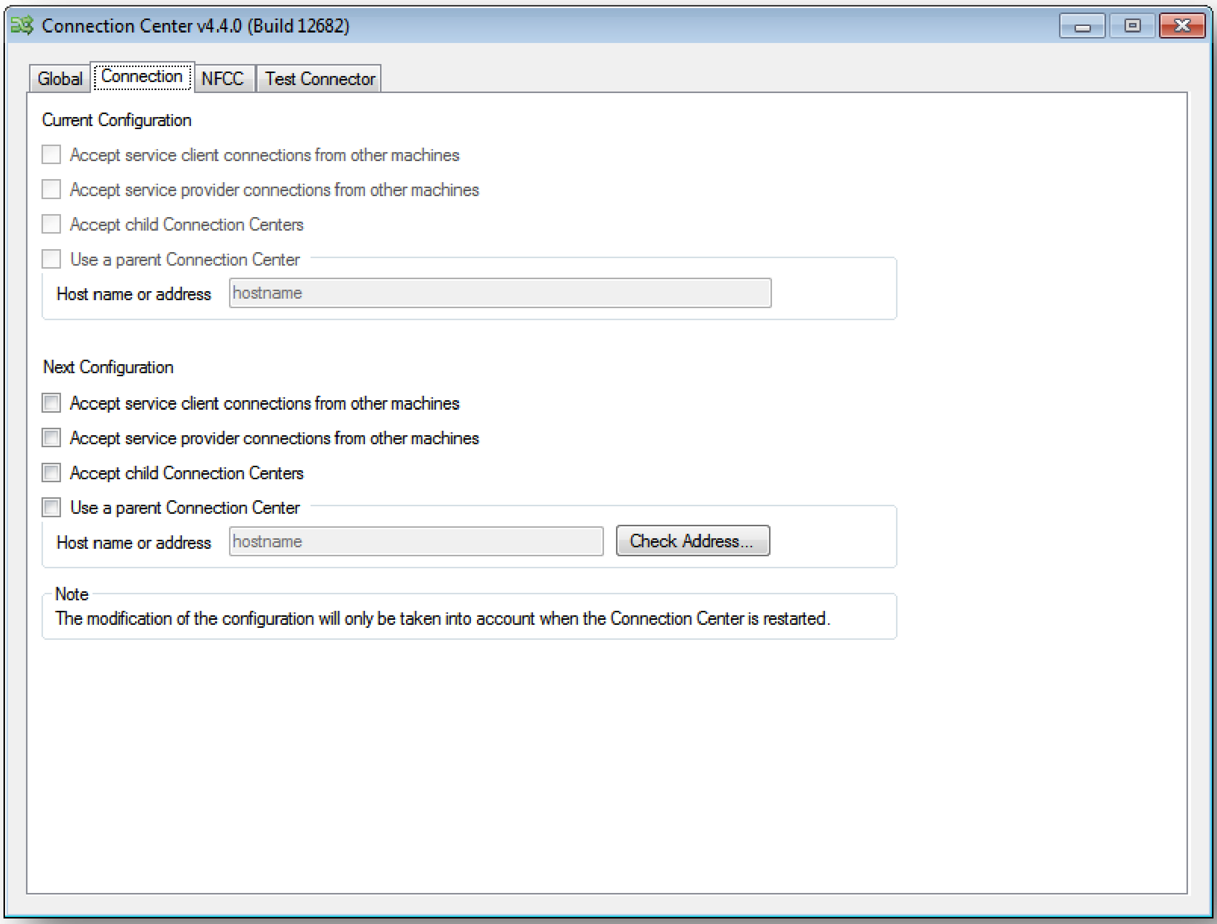
A helper is provided to retrieve Service Provider names, when a Service Client wants to connect to it:

- use the [Service List …] button to discover the existing Service Providers
- right-click on the service of interest; the “Copy Service URI” textbox will appear
- click onto the “Copy service URI” textbox, and the full URI of the Service Provider is pasted to the Clipboard
- paste the clipboard where appropriate.


Service list
Services
INSIDE1453
NfcSimulator
RF Interface
RF Device 1
RF Device 2
RF Device 3
NFC Controller (NFC HAL)
NFC Device #1
NFC Device #2
Presentation Robot
Virtual Robot
Connection Center
NFC Controller (I2C)
NFC Controller (I2C) on Aa
Copy service URI
Refresh
By Service
OK

3.3 Using the [Connection] Tab

Use this tab to customize the way the Connection Center manages connections with remote machines, or with virtual machines.


Connection Center v4.4.0 (Build 12682)

Global | Connection | NFCC | Test Connector

Current Configuration

- [ ] Accept service client connections from other machines
- [ ] Accept service provider connections from other machines
- [ ] Accept child Connection Centers
- [ ] Use a parent Connection Center

Host name or address: hostname

Next Configuration

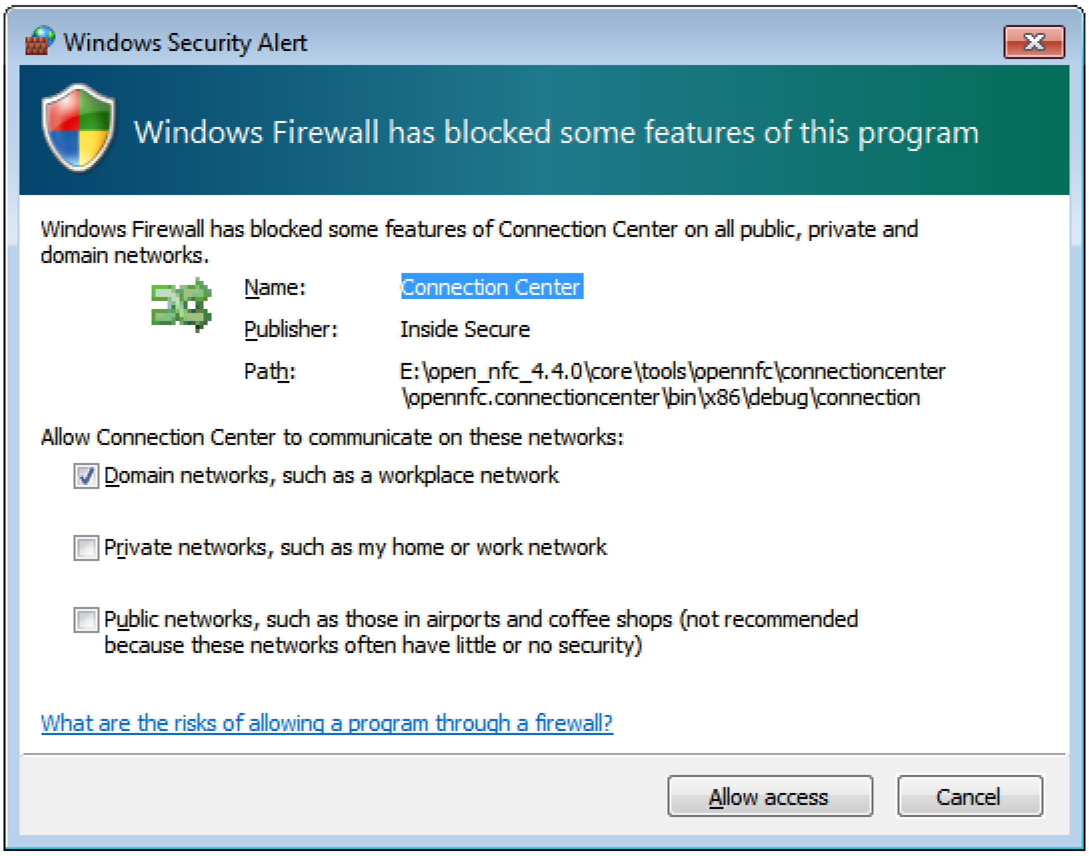
- [ ] Accept service client connections from other machines
- [ ] Accept service provider connections from other machines
- [ ] Accept child Connection Centers
- [ ] Use a parent Connection Center

Host name or address: hostname

Note
The modification of the configuration will only be taken into account when the Connection Center is restarted.

Note on Windows security center:

When you will enable connections for the first time, you will probably get a ***Windows Security Alert*** (depending on your security settings), like this:



Windows Security Alert

Windows Firewall has blocked some features of this program

Windows Firewall has blocked some features of Connection Center on all public, private and domain networks.

Name:	Connection Center
Publisher:	Inside Secure
Path:	E:\open_nfc_4.4.0\core\tools\opennfc\connectioncenter\opennfc.connectioncenter\bin\x86\debug\connection

Allow Connection Center to communicate on these networks:

- [✓] Domain networks, such as a workplace network
- [ ] Private networks, such as my home or work network
- [ ] Public networks, such as those in airports and coffee shops (not recommended because these networks often have little or no security)

What are the risks of allowing a program through a firewall?

If so, you will have to select the **Unblock** option to allow “Connection Center.exe” to accept TCP connections from the other machines.

<u>“Accept service client connections from other machines”</u>

This option lets you connect to remote service clients. For example, this option needs to be checked if you use a remote NFC Controller from a Linux-based PDA, accessible through a TCP/IP connection.

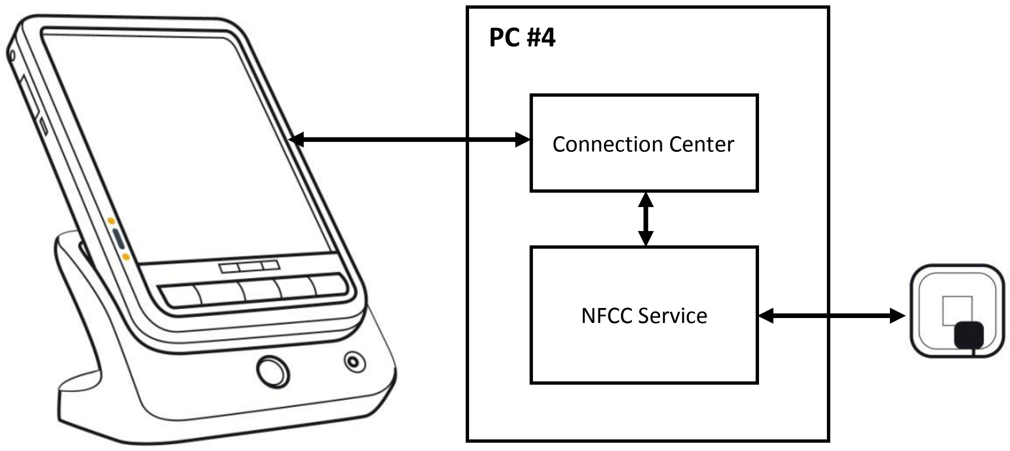


Figure 2 – Accept Service client connections from other machines

<u>“Accept service provider connections from other machines”</u>

This option lets you connect to remote service providers.

For example, this option must be checked when you want to allow a handset to connect to the Connection Center, in order to run tests offered by the Test Server. The handset acts as a “Test Agent” provider.

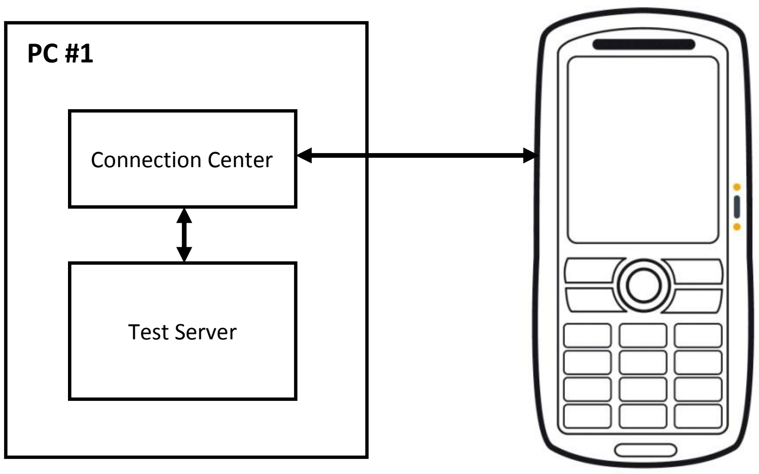


Figure 3 – Accept Service Provider connections from other machines

<u>“Accept child Connection Center”</u>

This option lets you connect to remote machines embedding a Connection Center, and declares your computer as the Master of the Connection Center tree.

The figure 1 is a typical illustration of this use case.

To decide if your computer is the Master, or one Slave, of the Connection Center tree is user choice, and should be decided depending on :

- PC availability (typically not set on a machine frequently rebooted, but instead on a non-development stable machine),
- PC visibility (typically should not be a Virtual Machine, hardly accessible from outside the host computer)

<u>“Accept a parent Connection Center”</u>

This option lets you connect to remote machines embedding a Connection Center, and declares your computer as one of the Slaves of the Connection Center tree.

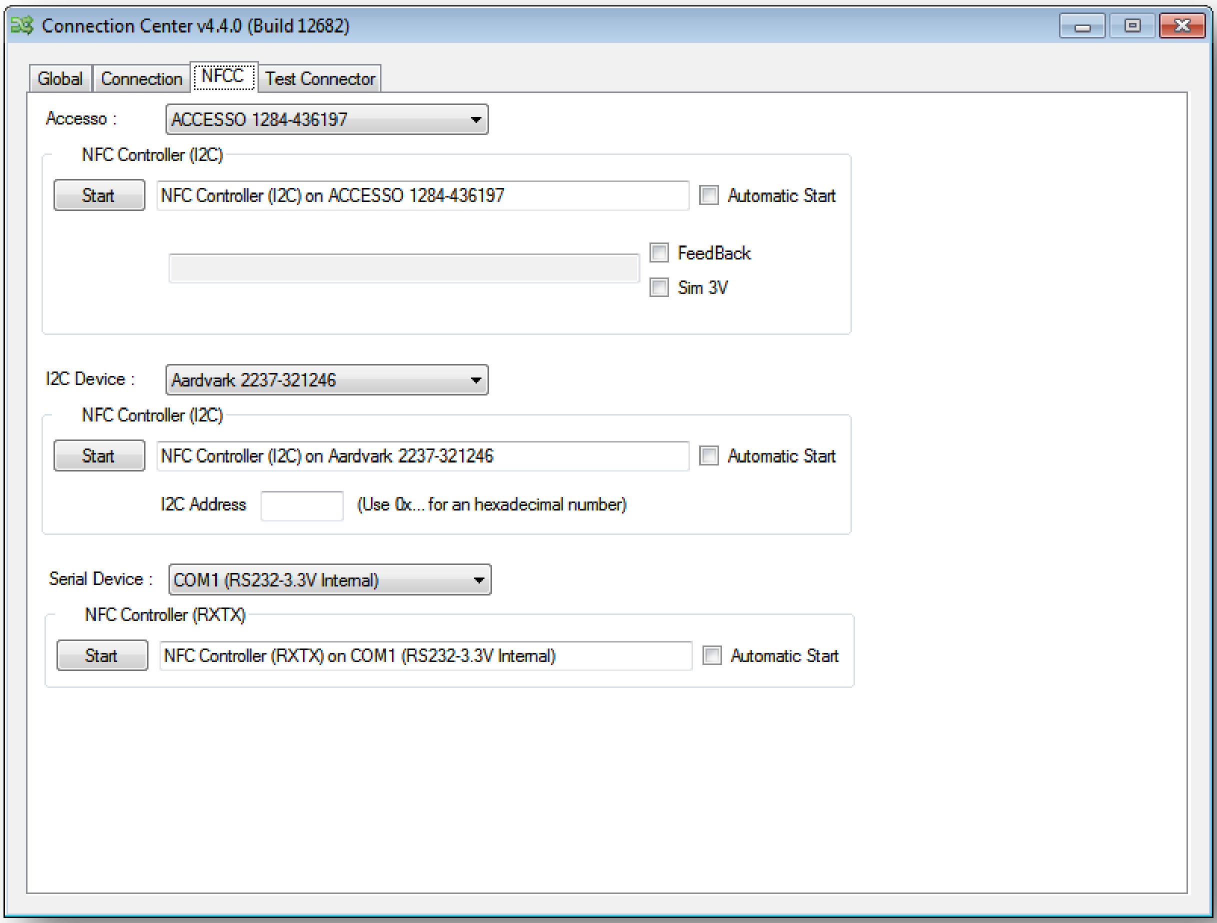
The figure 1 is a typical illustration of this use case.

3.4 Using the [NFCC] Tab

Use this Tab for connecting to a local NFC Controller.

Three NFC Controller types are predefined:

- I2C (through DeskRead/Accesso) ,
- I2C (through Aardvark adapter) ,
- RXTX (through CP210x USB/Serial adapter, or standard serial communication port)



Accesso : ACCESSO 1284-436197

NFC Controller (I2C)

NFC Controller (I2C) on ACCESSO 1284-436197 ☐ Automatic Start

☐ FeedBack

☐ Sim 3V

I2C Device : Aardvark 2237-321246

NFC Controller (I2C)

NFC Controller (I2C) on Aardvark 2237-321246 ☐ Automatic Start

I2C Address (Use 0x... for an hexadecimal number)

Serial Device : COM1 (RS232-3.3V Internal)

NFC Controller (RXTX)

NFC Controller (RXTX) on COM1 (RS232-3.3V Internal) ☐ Automatic Start

For both “NFC Controller (I2C)”, you must first select the I2C Device to connect. If you don’t give the i2C address then the default value (0x5e) is selected.
For DeskRead device you can:

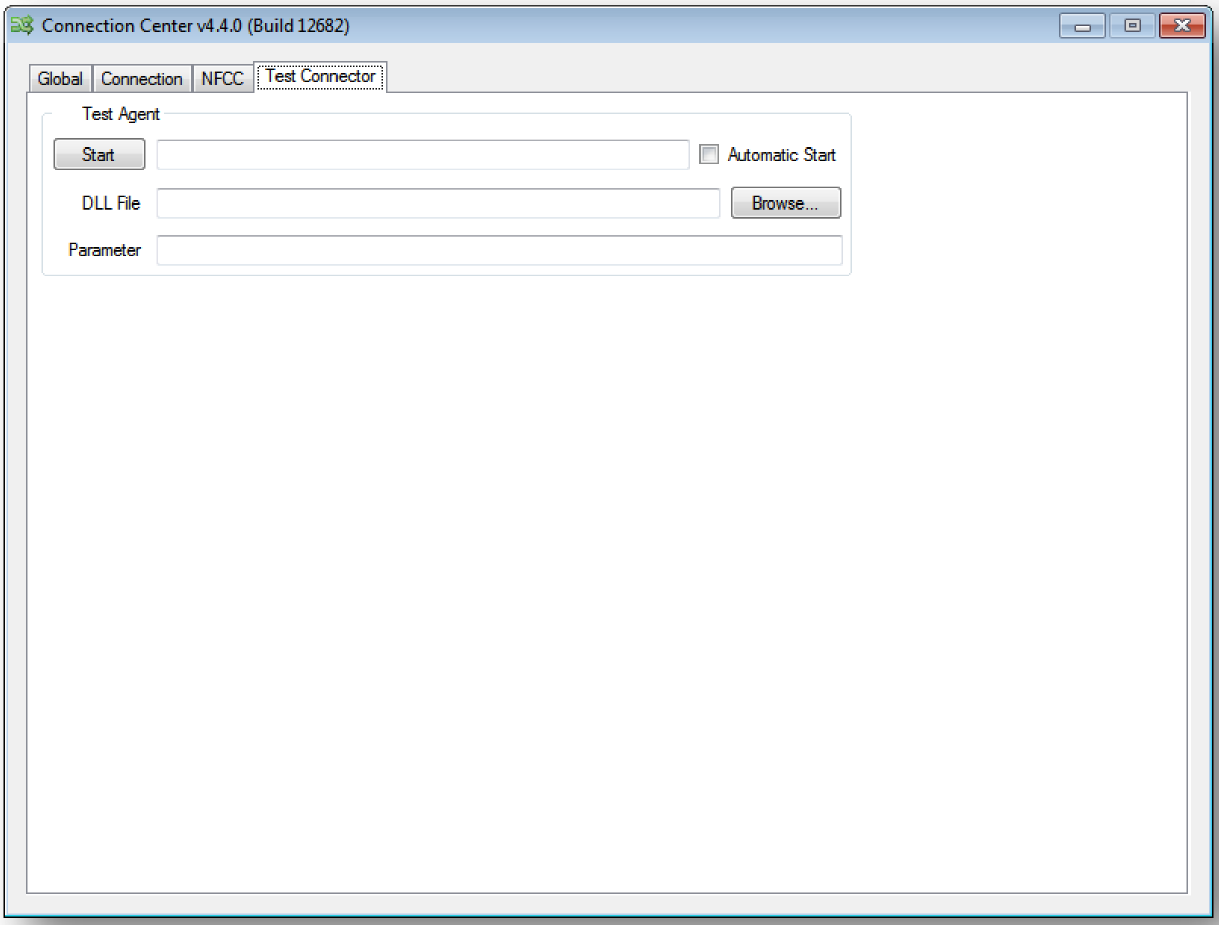
- retrieve some information by checking “FeedBack” option and
- Force the Sim 3V.

For the “NFC Controller (RXTX)” you must first select the Serial Device in the list of available serial devices, then press the button [Start] to connect to the local communication port.

If a new device is plugged, and if you want to refresh the list of available I2C Devices and Serial Devices, just move to another tab, then when you come back to the [NFCC] Tab, the list will be refreshed.

3.5 Using the [Test Connector] Tab

Use the Test Connector Tab to connect to a custom connector service.



In the “DLL File” TextBox, enter the name of your DLL that performs the Test Connector functionality. You can either enter the name directly or [Browse] to the file location.
If your DLL needs input parameters, just enter the command line in the TextBox “Parameters”.

When the DLL definition is completed, press [Start] button and the Connection Center will launch the DLL; if you want to load this DLL automatically when launching the Connection Center, check the Box “Automatic Start”.

3.5.1 Description of the "Test Connector"

For example, if you have a specific physical link to your target (else than TCP/IP or Serial), you have to develop a small Test Connector library, that will be loaded and used by the Connector Center to connect to your target.

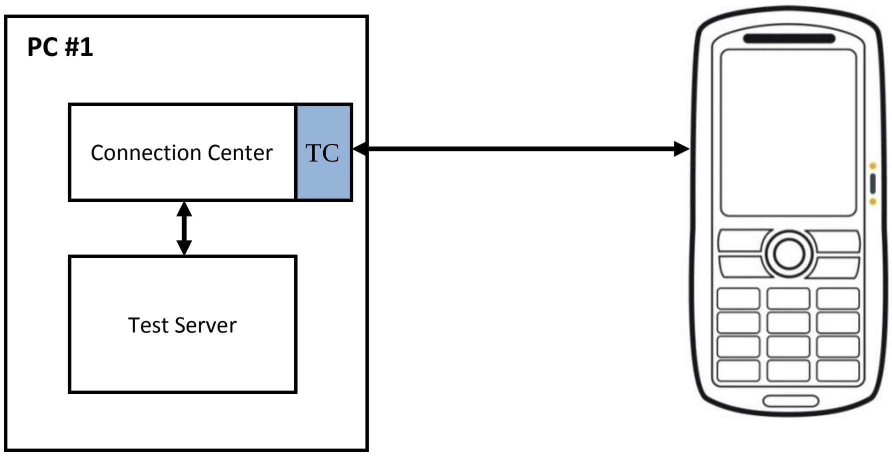


Figure 4 – Custom Connection to a Target

3.5.2 Implementing the "Test Connector"

The Test Connector is a set of two simple applications or libraries executed on the Test PC and on the target handset. The Test Connector transmits the messages from the Test Server to the Test Engine Message function. The Test Connector is developed during the porting of Open NFC. A test connector for the PC is needed only when no TCP/IP is available to make a connection to the target.

3.5.2.1 Principle

The protocol of the Test Server is a simple message based exchange. The Test server sends a message and waits for the answer before sending the next message. The Test Connector, on the PC or on the target, does not interpret any message. The message sent by the Test server should be transferred to the Test engine with the function WChipExchangeMessage(). The answer of the Test Engine is received by the callback function and should be transferred to the Test Server.

3.5.2.2 Custom Connector

To use any other protocol or physical link between the Test Engine and the target, a custom DLL connector should be developed to be plugged on the Test Engine. The functions implemented in this DLL are needed to open a connection, send & receive data, and close a connection.

Four functions should be included in the DLL, with their prototypes defined exactly as the ones below. A skeleton project is provided to build a Custom Connector DLL. The following file (CustomConnector.cpp) is part of the project.

```
/*
 * Skeleton project to build a custom connector.
 */

extern "C" __declspec(dllexport) int OpenDevice(char* pDeviceString);
extern "C" __declspec(dllexport) void CloseDevice(int hDevice);
extern "C" __declspec(dllexport) int Write(int hDevice, char* pBufferIn, int offset, int nBufferInLength);
extern "C" __declspec(dllexport) int Read(int hDevice, char* pBufferOut, int nBufferOutLength);

/**
 * @brief   Opens a Device using the device string URI.
 *
 * @param[in]  pDeviceString  String identifying the device.
 *
 * @return  a handle to the device.
 *          0 if the device could not be opened.
 **/
int OpenDevice(char* pDeviceString)
{
  return true;
}

/**
 * @brief   Sends data to the opened device.
 *
 * @param[in]  pBufferIn  The buffer containing the data to send.
 *
 * @param[in]  nBufferInLength  The length in bytes of the data to send.
 *
 * @return  The number of bytes actually written.
 **/
int Write(int hDevice, char* pBufferIn, int offset, int nBufferInLength)
{
  return nBufferInLength;
}

/**
 * @brief   Reads data from the opened device.
 *
 * @param[in]  pBufferOut  The buffer receiving the data.
 *
 * @param[in]  nBufferOutLength  The size in bytes of the buffer.
 *
 * @return  The actual length in bytes of the data read.
 **/
int Read(int hDevice, char* pBufferOut, int nBufferOutLength)
{
  return 0;
}

/**
 * @brief   Closes the connection.
 **/
void CloseDevice(int hDevice)
{
}
```

3.5.3 Implementing the "RS232 Connector"

3.5.3.1 Introduction

This section contains the documentation for the RS232 Test Connector. It describes the RS232Connector.dll Configuration and use. It also describes the RS232 Message format and must help a final user to write the device and the pc part of a Test Connector.

3.5.3.2 Overview

The RX/Tx plug-in Allow an RS232 connection between the Test Server and the Target Device.

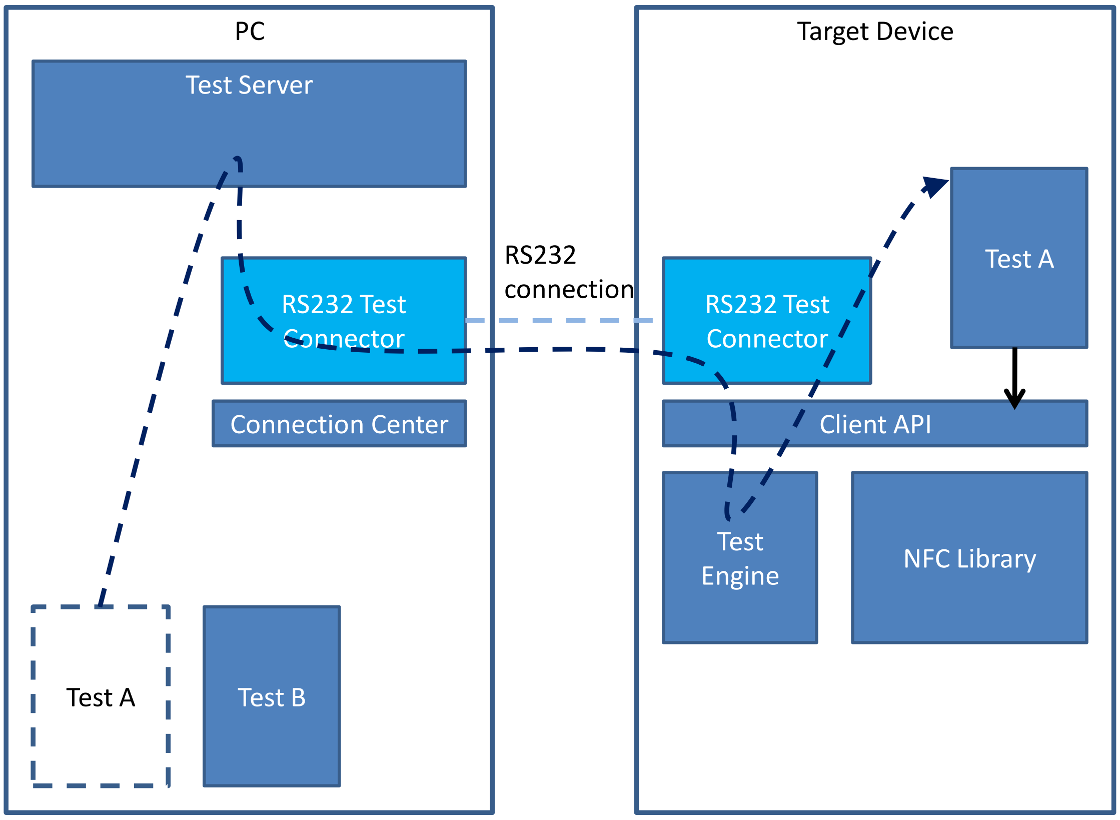
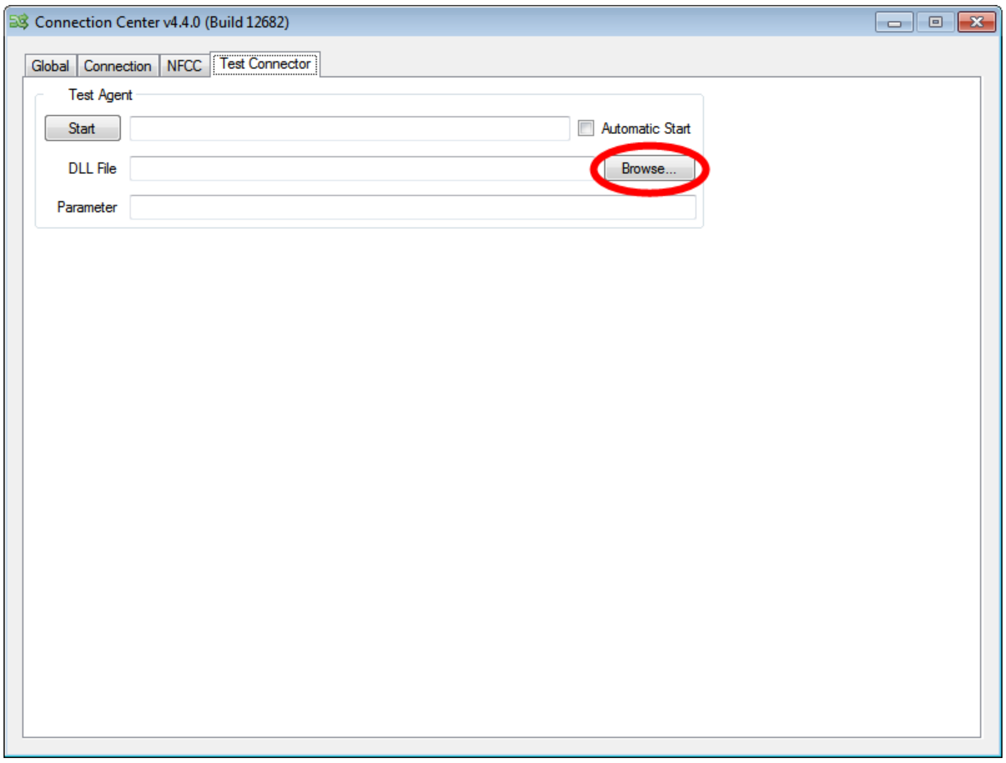


Figure 5: Test infrastructure overview for native tests with the RS232Connector

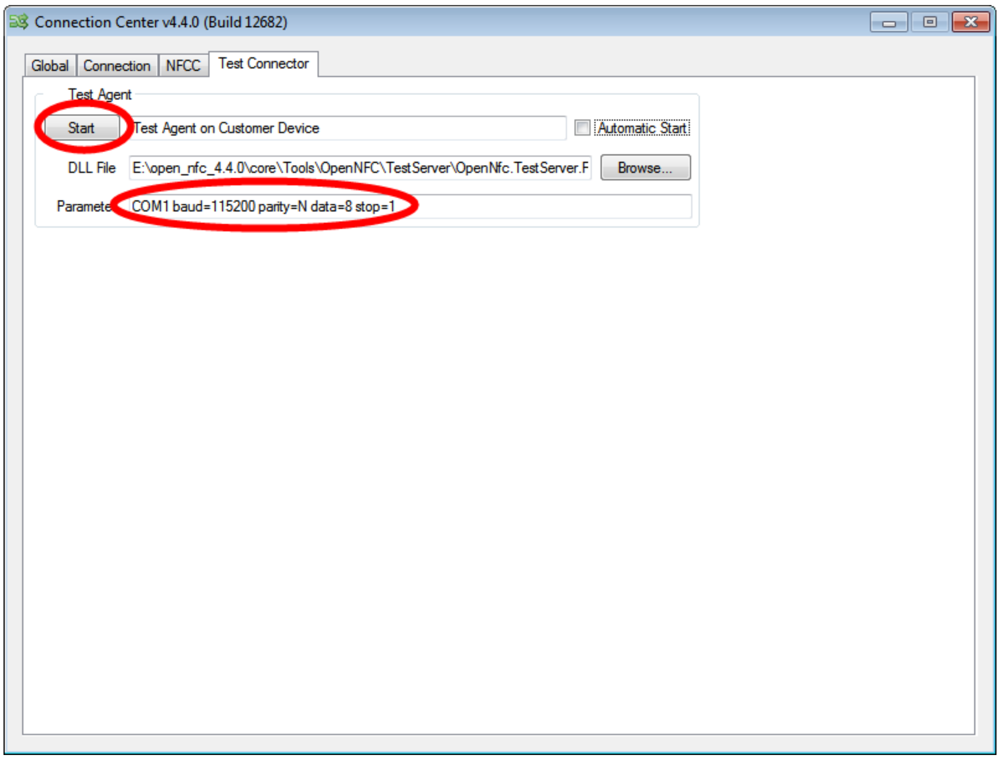
3.5.3.3 Connector configuration

In the Connection center windows select [Test Connector] Tab

Select the RS232Connector.dll



Type the RS232 serial port configuration string (depending of the target device) and click on start Button



Test Agent on Customer Device

DLL File: E:\open_nfc_4.4.0\core\Tools\OpenNFC\TestServer\OpenNfc.TestServer.F

Parameter: COM1 baud=115200 parity=N data=8 stop=1

The Parameter string has the following format:

COM*x* [baud=*b*] [parity=*p*] [data=*d*] [stop=*s*]
[to={on|off}][xon={on|off}][odsr={on|off}][octs={on|off}][dtr={on|off|hs}][rts={on|off|hs|tg}][idsr={on|off}]