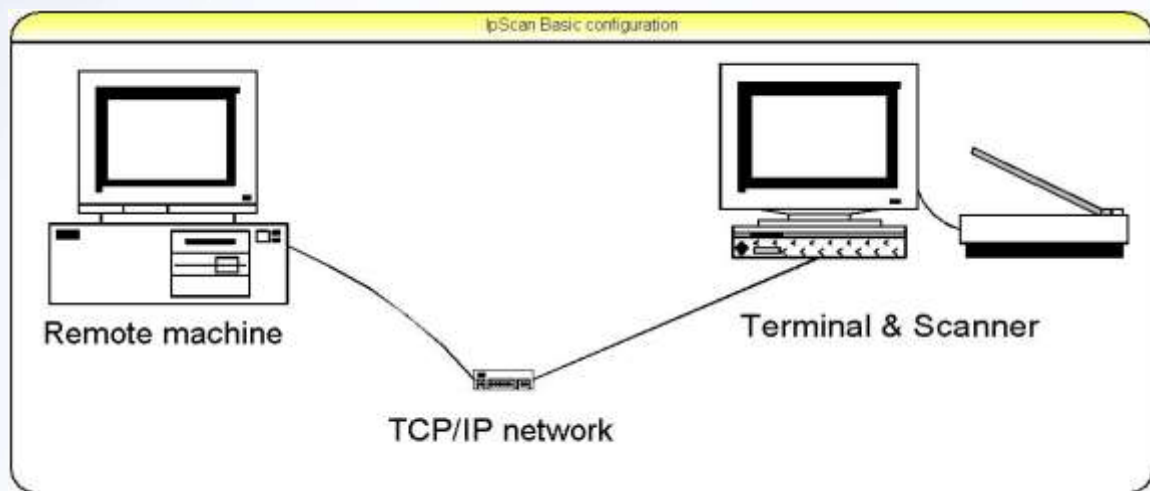


IPScan® Manual

1. General

IPScan® lets you control a scanner connected to a PC from a different PC in your network.

In most cases IPScan® will be installed on the terminal machine and the scanner will be controlled through it from different software using our SDK.



You will have to use the CSSN SDK for activating IPScan®.

From the SDK point of view IPScan® is just another scanner. Minor changes are required to activate it just as any other CSSN scanner.

2. Network configuration

IPScan® uses the TCP/IP protocol for the purpose of communicating with the controlling application.

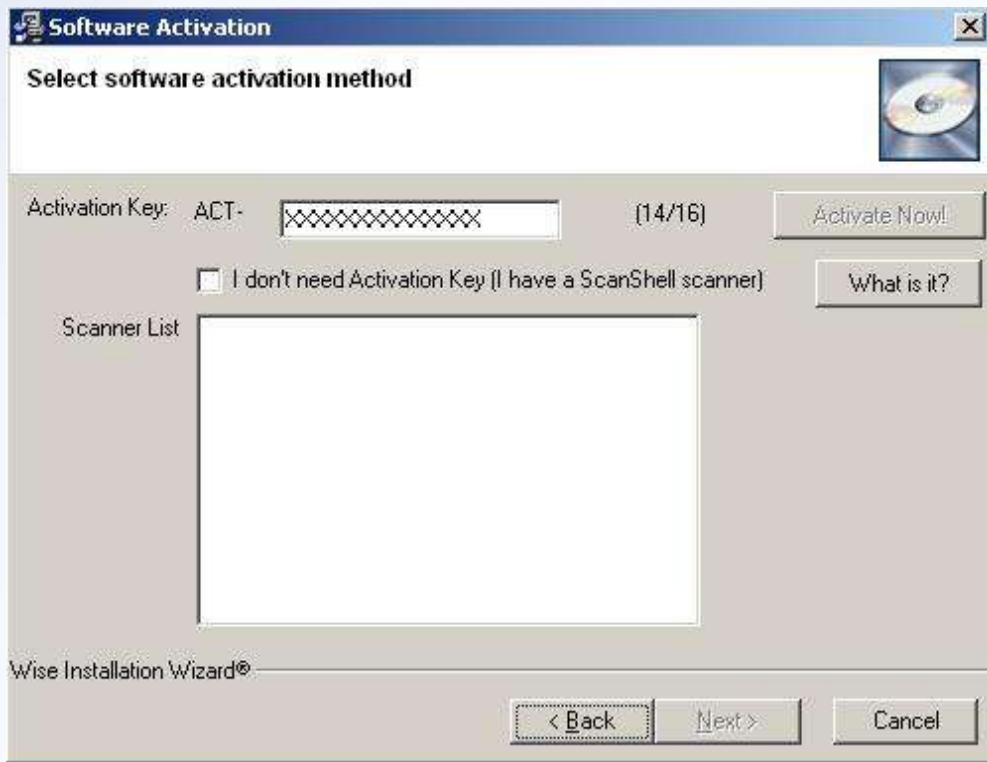
Currently, in version 2.0.1.4 IPScan® must have its own IP address. The controlling application must know this IP for establishing the connection.

In some cases the machine IPScan® is installed on is connected to the network using NAT (Doesn't have its own IP address outside your local network) you can use port forwarding (i.e. - configure your router to transfer any communication on the chosen port to the IPScan® machine). This way machines from outside your network will be able to connect to IPScan®.

IPScan® Manual

3. Setting up IPScan®.

Run the IPScan® setup program. On the software activation screen you will have to provide an activation key that is suitable for IPScan®.



On the port setup screen (see below) choose the desired port number; it can be any port you like as long it is not blocked by firewalls, antivirus, and so on.

The data port number is optional it could be the same as the main port. It is relevant only when working with asynchronous scans. The asynchronous scan function returns immediately after the scanner completed the scanning and does not wait for the image to arrive through the network. The regular scanning waits until the image arrives to the controlling machine. Detailed usage is in the SDK documentation.

IPScan® Manual





4. Using IPScan®


After IPScan® is installed it will be running in the background and will wait for incoming connections.




The icons point out the state of IPScan®:

1. IPScan® license is not valid. 
2. IPScan® is not connected to a controller. 

IPScan® Manual

3. IPScan® is connected but not attached to a scanner. 

4. IPScan® is connected and the scanner attached. 

Right clicking on the icon will open a pop up menu.

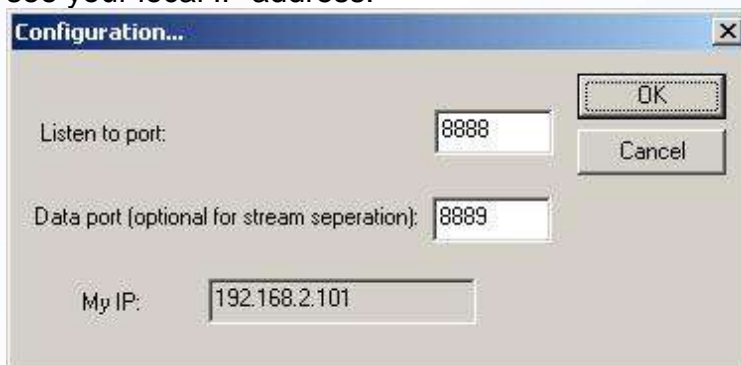


About – lets you see the current version installed on your machine.



Scan test - For testing purposes only, use it to test the scanner manually, it will work only if a controller is connected to IPScan®.

Configuration – Lets you change the port number IPScan® listens to, and see your local IP address.



Close – close the application.

IPScan® Manual

5. Connecting to IPScan® from the SDK.

The best way to use IPScan® is integrating it in your software using our proprietary SDK. The SDK functions are described in the files VC_SDK.pdf and ScanWex.pdf. The SDK comes with samples in most of the programming languages.

Below is a sample C# code that demonstrates the usage of the SDK with IPScan®:

```
// Declaration
private SCANWLib.SLibExClass mSDK_SLib;
private SCANWEXLib.SLibExClass mSDK_SLibEx;
private SCANWLib.IdDataClass mSDK_IDData;

// creating the library
mSDK_SLib = new SCANWLib.SLibExClass();
mSDK_SLibEx = new SCANWEXLib.SLibExClass();
mSDK_IDData = new SCANWLib.IdDataClass();

// Init
mSDK_SLibEx.InitLibraryEx("MyLicenseNumber", 0, "MyApplication");
mSDK_SLib.InitLibrary("MyLicenseNumber ");
mSDK_IDData.InitLibrary("MyLicenseNumber ");

// Ask the SDK to connect to IPScan® with a default scanner

mSDK_SLibEx.UseFixedModel = (short)SCANNERS.CSSN_IP; // CSSN_IP the
// Give the remote IP address and the port number to the library
mSDK_SLibEx.SetRemoteIP(IPText.Text, nPort,
(short)SCANNERS.CSSN_NONE);

// Make sure that the scanner is connected
int sValid = mSDK_SLib.IsScannerValid;

if (sValid == 0)
{
    MessageBox.Show("Scanner not found");
    return;
}

// Scan the file
mSDK_SLib.ScanToFileEx ("C:\\myFile.jpg");

// Process the ID data exactly you would do it with a local scanner

int stateId;
int retVal;

stateId = mSDK_IDData.AutoDetectState("");
retVal = mSDK_IDData.ProcState("", stateId);
retVal = mSDK_IDData.RefreshData();
```