Examples 1a, 1b, 1c: Finding Scanners on the Network

Scanner discovery follows a request/response model, whereby the user's application requests scanners that meet certain criteria (it can also request all available scanners), and the scanner(s) respond to the request by sending their ID information. The responses are not guaranteed to be returned in real-time; they're sent via UDP. Consequently, after the request, the functions used for discovering scanners block for approximately one second to reasonably ensure that all scanners get their responses in.

> **Best Practice**
>
> In general, we recommend that you do not use the discovery functions in a production application. Instead, it is better to configure your scanners once and store IP addresses and CableIds with your application (e.g., in a settings file or database). For diagnostics and configuration applications, the usage of the discovery functions is fine.

The available discovery functions are located in the class ScannerConfig:

- public static List<DiscoveryResponse> ScannerConfig.FindAllScanners() (Example 1a)
- public static DiscoveryResponse ScannerConfig.FindScanner(int id) (Example 1b)
- public static void ScannerConfig.FindAllScanners(ScannerResponseDelegate srh) (Example 1c)

Discovering All Scanners – Example 1a

To discover all scanners on a network, use the following function:

- public static List<DiscoveryResponse> ScannerConfig.FindAllScanners()

```
using System.Collections.Generic;
using JoeScan.JCamNet;

namespace Examples
{
    class Program
    {
        static void Main(string[] args)
        {
            List<DiscoveryResponse> responses = ScannerConfig.FindAllScanners();
            foreach (var discoveryResponse in responses)
            {
                System.Console.WriteLine("Base IP Address: {0}, CableId: {1}, Serial Number: {2}",
                    discoveryResponse.BaseIPAddress,
                    discoveryResponse.CableId,
                    discoveryResponse.SerialNumber );
            }
        }
    }
}
```

Sample Output:

```
Base IP Address: 192.168.1.105, CableId: 0, Serial Number: 374
Base IP Address: 192.168.1.150, CableId: 0, Serial Number: 1465
Base IP Address: 192.168.1.105, CableId: 22, Serial Number: 1941
Base IP Address: 192.168.1.105, CableId: 28, Serial Number: 1940
Base IP Address: 192.168.1.105, CableId: 30, Serial Number: 1747
```

Querying for a specific scanner by CableId or Serial Number – Example 1b

If you want to query for a specific scanner, either by its CableId or serial number, use the following function:

- public static DiscoveryResponse ScannerConfig.FindScanner(int id)

The parameter **id** is either a CableId or a serial number. Because no hardware with serial numbers below 128 exists, the API assumes any argument < 128 is a CableId, and a serial number otherwise. However, there may be multiple scanners on your network that have the same CableId (and a different IP Base Address). In this case, it is not safe to use this function, as it only returns one DiscoveryResponse and may throw an exception.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using JoeScan.JCamNet;

namespace Examples
{
    class Program
    {
        private static int serialNumber = 1465;
        private static int cableId = 0;

        static void Main(string[] args)
        {
            DiscoveryResponse response = ScannerConfig.FindScanner(serialNumber);
            System.Console.WriteLine("Base IP Address: {0}, CableId: {1}, Serial Number: {2}",
                    response.BaseIPAddress, response.CableId, response.SerialNumber);

           // don't do this: if multiple scanners respond, you will receive an exception
           // of type JoeScan.JCamNet.ScannerCommunicationException

           // response = ScannerConfig.FindScanner(cableId);
           // System.Console.WriteLine("Base IP Address: {0}, CableId: {1}, Serial Number: {2}",
           // response.BaseIPAddress, response.CableId, response.SerialNumber);

        }
    }
}
```

Sample Output:

```
Base IP Address: 192.168.1.150, CableId: 0, Serial Number: 1465
```

Finding scanners by using specific criteria – Example 1c

You can also find scanners by using a delegate method of type ScannerResponseDelegate, which is called for each DiscoveryResponse received from the network. This is useful when you have specific criteria to build the list of scanners you want to address (e.g., all scanners on a specific subnet).

Tip: For a more detailed explanation of subnet testing, see http://en.wikipedia.org/wiki/Subnetwork

In the following example, we only want to work with scanners that have a "1" in their network address at the 3rd quad. Use the following function:

- public static void ScannerConfig.FindAllScanners(ScannerResponseDelegate srh)

```
using JoeScan.JCamNet;
using System;

namespace Examples
{
    class Program
    {
        static void Main(string[] args)
        {
            ScannerConfig.FindAllScanners(MyDelegate);
        }

        public static void MyDelegate(DiscoveryResponse response)
        {
            // filter here by your criteria, for instance the subnet:
            if (response.BaseIPAddress.GetAddressBytes().GetValue(2).ToString() == "1")
            {
                System.Console.WriteLine("Base IP Address: {0}, CableId: {1}, Serial Number: {2}",
                    response.BaseIPAddress, response.CableId, response.SerialNumber);
            }
        }
    }
}
```

Sample Output:

```
Base IP Address: 192.168.1.150, CableId: 0, Serial Number: 1465
Base IP Address: 192.168.1.105, CableId: 22, Serial Number: 1940
Base IP Address: 192.168.1.105, CableId: 30, Serial Number: 1941
```