



SDK9000 - Software development kit for EchoStream wireless devices

The Inovonics software development kit (SDK) greatly simplifies the interface between custom application software and Inovonics EchoStream wireless devices. Using the SDK interface library, developers are no longer burdened with the raw detail of EchoStream message formats, bit fields, and RF network operations — saving days, weeks, even months of software development time. As a result, your organization will develop better wireless enabled solutions, in less time, at lower product development costs.

The Inovonics SDK is designed with the Microsoft® .NET™ developer in mind. Here's a sample of what developers are saying:

"With no prior knowledge of EchoStream or the Inovonics SDK, our software was integrated into an EchoStream network in less than two hours, something that would have otherwise taken weeks."

– Scott Thomas, Principal Software Engineer, Cerebral Syntax, Boulder, CO

The SDK supports a variety of EchoStream wireless products. The SDK includes support for wireless pendants, contact closures, environmental sensors, and utility measurement devices, making it easy to integrate a wide range of life safety, security, temperature, humidity, and utility consumption applications into your custom software.

The SDK includes the following components:

- EchoStream interface dynamic linked library (the SDK core, referenced from within the Microsoft .NET project)
- Network configuration utility (used on site to configure EchoStream wireless network infrastructure)
- Demonstration applications (provide examples of SDK implementation)
- Developer guides, user manuals, and quick reference tools

An evaluation version is available at no charge. The evaluation version of the SDK provides all of the functionality of the full version, but it has a runtime limit of 30 minutes. Contact the Inovonics sales team (800-782-2709 or sales@inovonics.com) to obtain a copy, or to purchase a fully functional Inovonics software development kit.

Inovonics SDK Technical Specifications

Development Environment:	Microsoft Visual Studio® 2005 or later
Prerequisite Software:	Microsoft Windows® Installer 3.1, Microsoft .NET framework version 2.0, Microsoft SQL Server™ 2005 express edition
Developer Expertise:	Strong working knowledge of Microsoft .NET framework, and familiarity with use of generics, interfaces, key/value pairs, dictionaries, asynchronous events, callbacks, and exceptions.
Supported Operating Systems:	Microsoft Windows Server® 2003 Standard Edition (32-bit x86); Microsoft Windows Vista®; Microsoft Windows XP Service Pack 2; Microsoft Windows 2000 Service Pack 4
System Requirements:	Intel® Pentium® 4 processor or equivalent 512 MB of RAM recommended, 10 MB of available hard disk space

Sample Code

```
// Description: A simple SDK usage example

using System;
using System.Collections.Generic;
using System.Threading;
using System.Text;

using Inovonics.EchoStream.Framework.User;

namespace QuickTest
{
    class Program
    {
        // Process a received message, displaying salient parts
        static void MsgEvent(INetwork net, IEvent evt)
        {
            IMessage msg = (IMessage)evt{NetworkNames.EventFields.MESSAGE};
            Console.WriteLine(
                "Rcv'd msg from S/N {0:D8}, {1}, status: {2}{3}{4},
                (int)msg[MessageNames.Fields.SRC_SER_NUM],
                ((string)msg[MessageNames.Fields.PRODUCT]).PadLeft(10),
                (bool)msg[MessageNames.Fields.TAMPER] ? "T" : " ",
                (bool)msg[MessageNames.Fields.LOW_BAT] ? "B" : " ",
                (bool)msg[MessageNames.Fields.RESET] ? "R" : " ";
            }

        // Setup and start the interface library running.
        static void Main(string[] args)
        {
            // Setup a driver.
            IDriver drv = Create.Driver(DriverNames.Kinds.SERIAL_PORT);
            drv[DriverNames.Options.DEVICE] = "COM1";
            drv.Open();

            // Probe the gateway.
            IProbe prb = Create.Probe(drv, true);
            String gwType = prb.Type;
            Prb.Stop();

            // Setup the network.
            INetwork net = Create.Network(gwType, drv, false);
            Net.RegisterCallback(NetworkNames.Events.MSG_RCVD,
                new NetworkEventHandler(MsgEvent));

            // Start the network up and watch it.
            net.Start();
            while (net.Running) { Thread.Sleep(10); }
        }
    }
}
```

Sample Code Output

```
Rcv'd msg from S/N 02505588, ES1223s, status: B
Rcv'd msg from S/N 02621142, ES1212, status:
Rcv'd msg from S/N 00526238, ES1210, status: T R
Rcv'd msg from S/N 02632245, ES1501, status: T
Rcv'd msg from S/N 01018952, ES1210w, status: T B
```