



Telescope

Free version 1.0.0

I. About the solution

Telescope is a tool for remote browsing of objects in a .NET (C#, VB.NET, ...) application you have developed. The main purpose for its creation is to give developers a the possibility of checking objects' states without the necessity of having to stop the run. This will allow developers to increase agility of their design process and to find and fix bugs in software quickly, which in turn will make the applications more robust, easy to use and reliable.

II. License

The basic version of Telescope 1.0.0 is distributed without a license, which means you have the right to use it without restriction for personal, educational or business purposes.

III. Installation

The installer file **Telescope.msi** deploys a set of files which are necessary in order to use Telescope on your machine. The list of files contains **Telescope.exe**, which aggregates data from nodes, assembly **TelescopeNode.dll** (see detailed description below) and current instruction. To delete Telescope from your computer execute the 'uninstall' function from Control Panel: Control Panel\Programs\Programs and Features

IV. Using Telescope

TelescopeNode.dll is an assembly responsible for reflecting information about state of objects inside your .NET application.

Telescope.exe is a program responsible for aggregation of data from nodes in your applications and displaying data in browser.

It is compatible with .NET 4.0 and higher.

To make binding to state of object you want to track in runtime, one should create instance of **Telescope. TelescopeNode** (node), passing reference to object as parameter in constructor. Other constructor parameters are name of the node and name object as they are displayed in browser.

An example of such a binding is depicted below:

```

namespace TestTelescopeConsole
{
    class Program
    {
        static void Main(string[] args)
        {
            Telescope.TelescopeNode n = new Telescope.TelescopeNode(new TestObject(), "NoName", "NoObjectName");

            Console.ReadLine();
        }
    }

    public class TestObject...
    public class InnerObj...
    public struct Str...
}

```

In case of binding to visual objects inside the WPF application (they can be accessed from window thread only), reference to System.Windows.Threading.Dispatcher object should be used as an additional parameter of node constructor. In this case, the corresponding code code appears as depicted in the next picture below:

```

namespace TestWPFApplication
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }

        Telescope.TelescopeNode node;

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            Dispatcher.Invoke(new Action(() =>
            {
                node = new Telescope.TelescopeNode(this, "ExampleNode", "ExampleObject", Dispatcher);
            })), null);
        }
    }
}

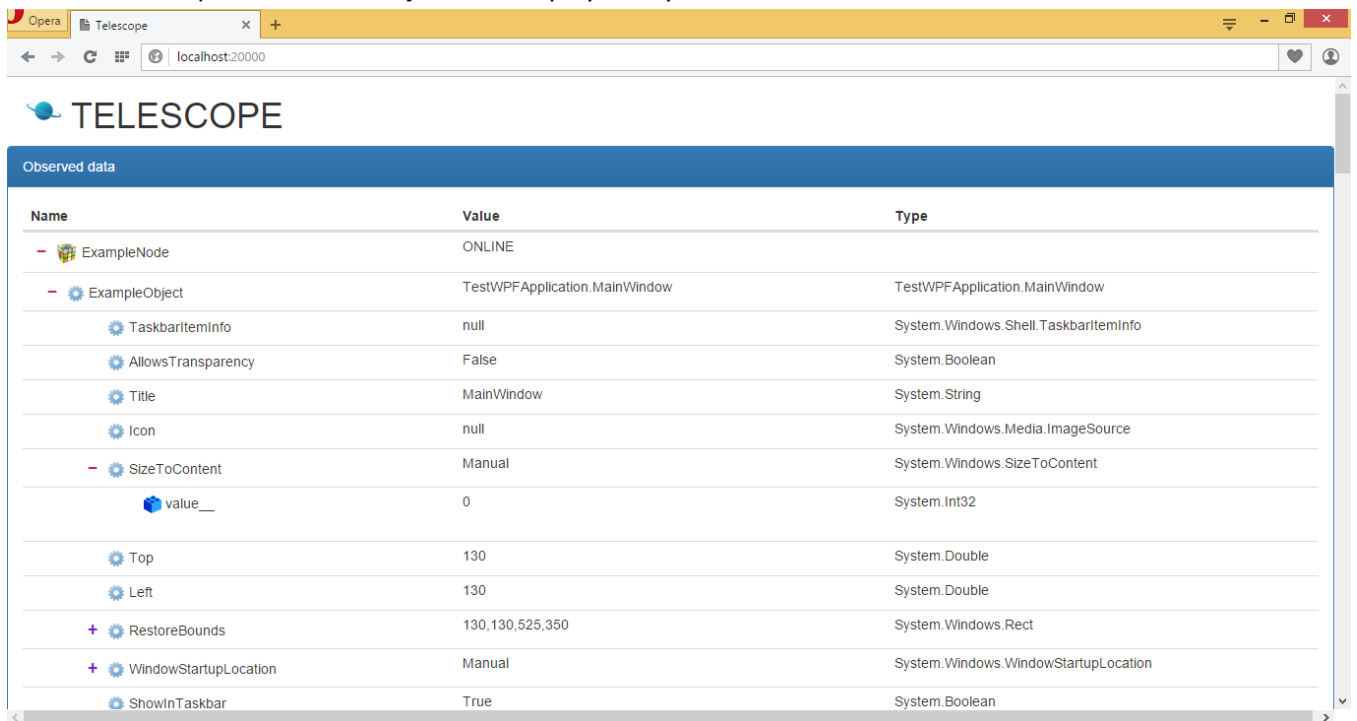
```

After you start your application, the created instance of **Telescope.TelescopeNode** will have access to the object to which it is bound and can transmit data to **Telescope** program. **Telescope** is capable of processing http requests it receives from the browser, and then sending valid html code with object data in response.

Telescope uses either the port you set in the configuration file **TelescopeConfig.cfg** (or port 20000 by default). This configuration file should be located in <Telescope folder>\config folder. Instance of **Telescope**. **TelescopeNode** stays inactive until **Telescope** sends it command to start the operation. **Telescope**. **TelescopeNode** sends requests to Telescope using port number set in configuration file **TelescopeNodeConfig.cfg** (or 20000 by default), which should be located in <Your application folder>\config. When the node connects Telescope, it dynamically assigns the port number to the node. The range of port numbers for nodes is set **TelescopeConfig.cfg** and is 20001-21000 by default.

If your browser and Telescope are located on the same computer, then the request you may use to browse your objects is <http://localhost:20000/>

Here is an example of how the objects are displayed in your browser:



Name	Value	Type
- ExampleNode	ONLINE	
- ExampleObject	TestWPFApplication.MainWindow	TestWPFApplication.MainWindow
TaskbarItemInfo	null	System.Windows.Shell.TaskbarItemInfo
AllowsTransparency	False	System.Boolean
Title	MainWindow	System.String
Icon	null	System.Windows.Media.ImageSource
- SizeToContent	Manual	System.Windows.SizeToContent
value__	0	System.Int32
Top	130	System.Double
Left	130	System.Double
+ RestoreBounds	130,130,525,350	System.Windows.Rect
+ WindowStartupLocation	Manual	System.Windows.WindowStartupLocation
ShowInTaskbar	True	System.Boolean

The screenshot shows the Telescope web application interface. At the top, there's a header with the Telescope logo and the text "TELESCOPE". Below that, a section titled "Observed data" contains a table with three columns: "Name", "Value", and "Type". The table lists various objects and their properties, including "NoName" (ONLINE), "NoObjectName" (TestTelescopeConsole.TestObject), "someIntData" (System.Int32[]) with sub-properties like Length, LongLength, Rank, SyncRoot, IsReadOnly, IsFixedSize, and IsSynchronized, and a "collection" (System.Int32[]) with elements [0], [1], and [2].

Name	Value	Type
- NoName	ONLINE	
- NoObjectName	TestTelescopeConsole.TestObject	TestTelescopeConsole.TestObject
- someIntData	System.Int32[]	System.Int32[]
Length	1000	System.Int32
LongLength	1000	System.Int64
Rank	1	System.Int32
+ SyncRoot	System.Int32[]	System.Int32[]
IsReadOnly	False	System.Boolean
IsFixedSize	True	System.Boolean
IsSynchronized	False	System.Boolean
- collection	System.Int32[]	System.Int32[]
[0]	0	System.Int32
[1]	1	System.Int32
[2]	2	System.Int32

Telescope displays a tree of public properties and fields of the objects. **In the basic version, the maximal depth of the tree is limited and equals 5.**



- depicts root of displayed node



- depicts root object or property of the object



- depicts field of the object



- depicts root of a collection

In case of loss of connection between node and Telescope program (for example, your application has been stopped) the node is marked as OFFLINE. An example of such a node is shown below:

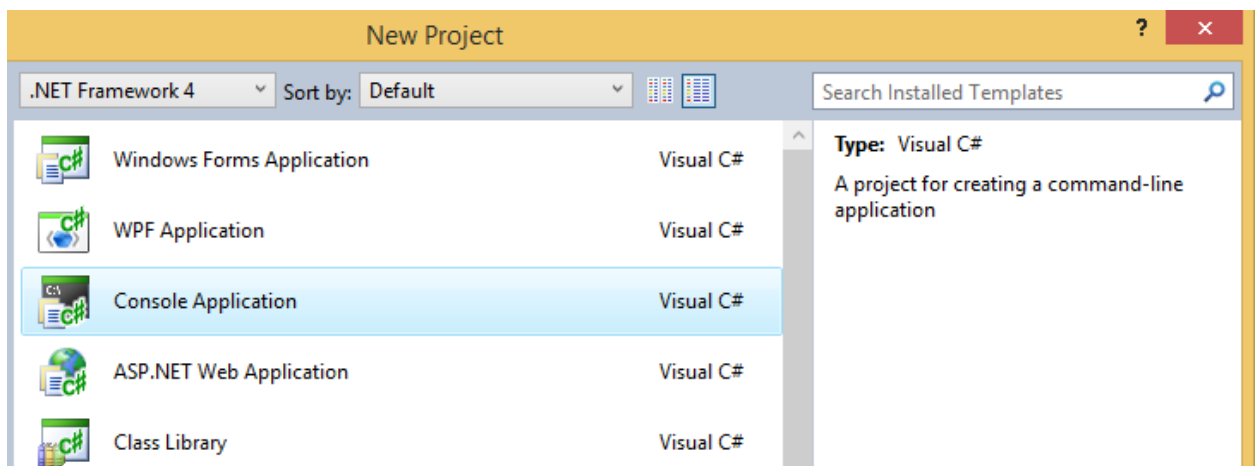
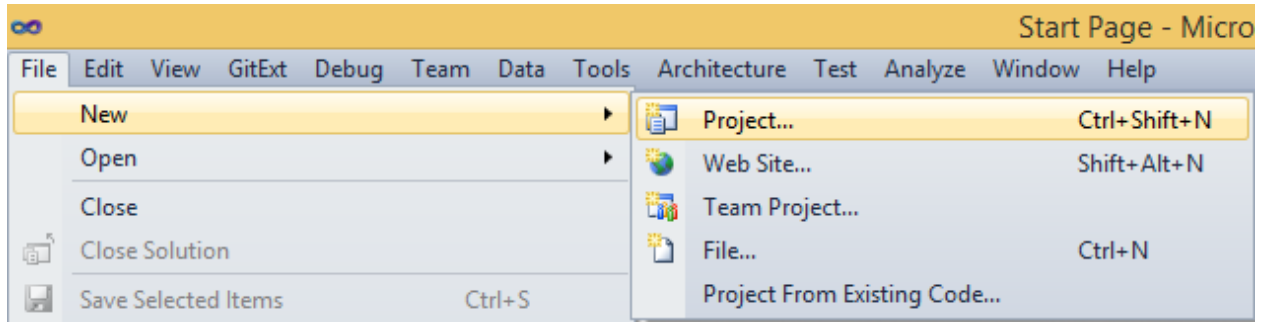
The screenshot shows the Telescope web application interface. The "Observed data" section contains a table with three columns: "Name", "Value", and "Type". The table lists "ExampleNode" (ONLINE) and "PORT: 20002" (OFFLINE).

Name	Value	Type
+ ExampleNode	ONLINE	
PORT: 20002	OFFLINE	

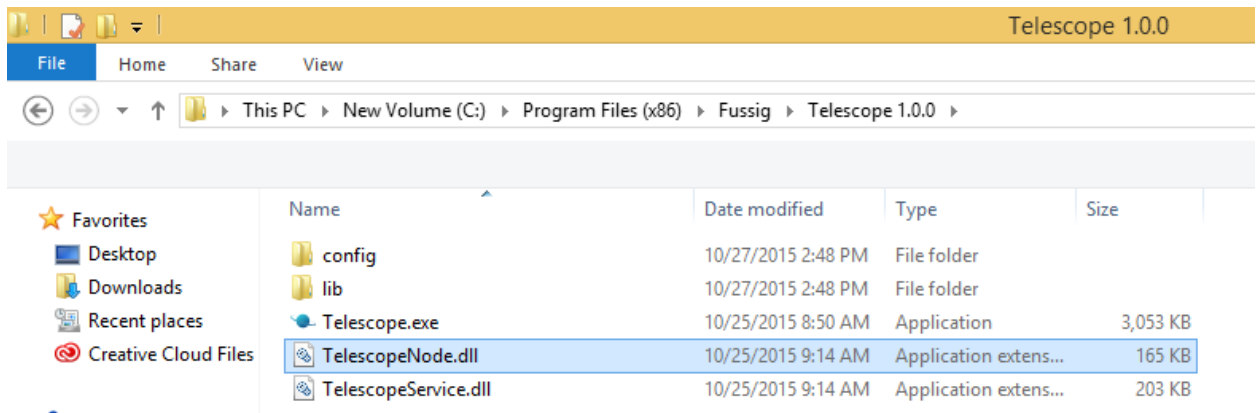
V. Examples

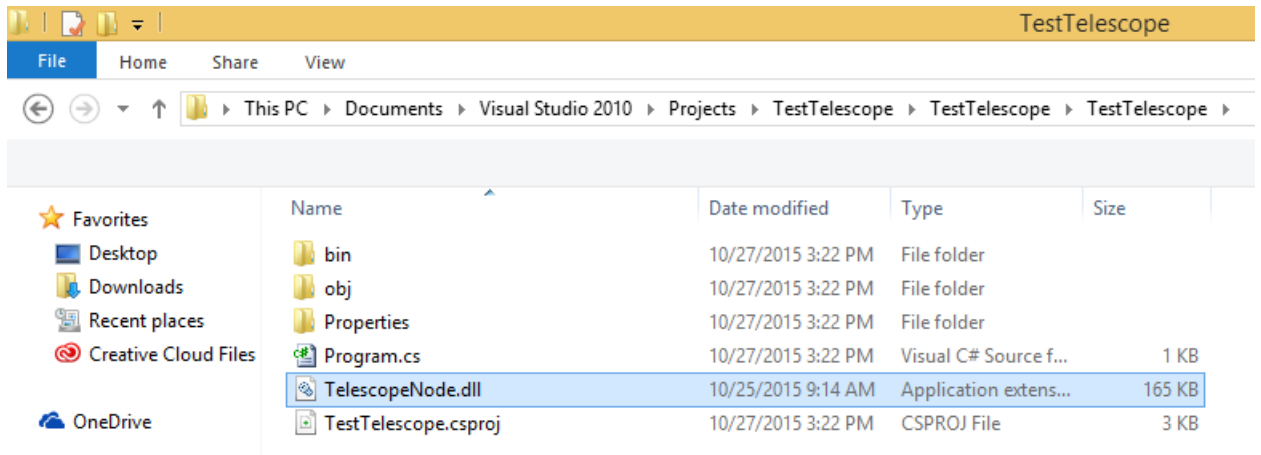
Let's consider the use of Telescope in a simple console program. Below are the step-by-step instructions for creating the application with TelescopeNode.

1. Create new console project in Visual Studio

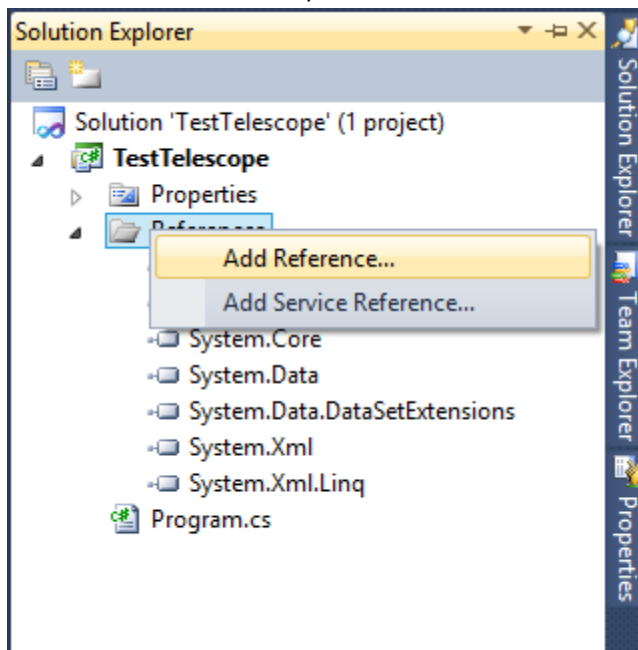


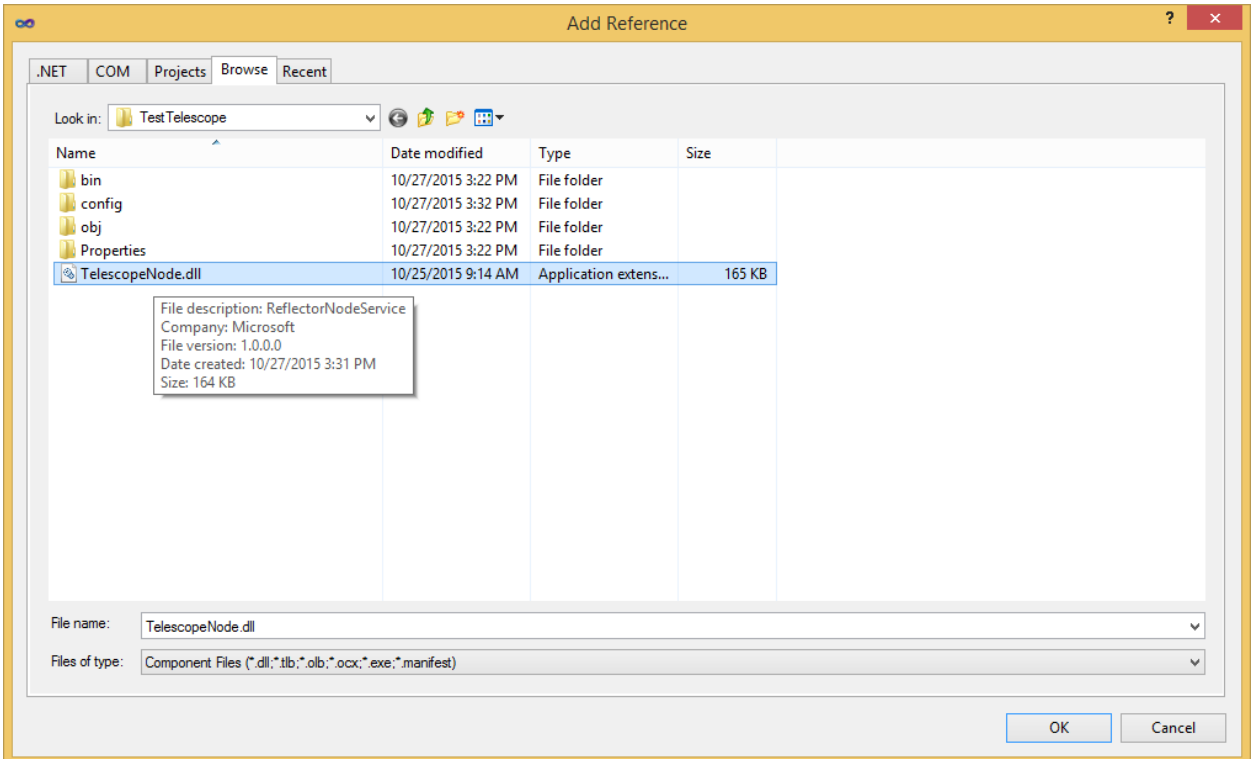
2. Copy **TelescopeNode.dll** from Telescope installation folder to your application folder (by default it is: %ProgramFiles%\Flussig\Telescope 1.0.0)





3. Add reference to TelescopeNode.dll





- Define objects you want to browse during runtime and make instance of TelescopeNode to make binding to them

```

namespace TestTelescope
{
    class Program
    {
        static void Main(string[] args)
        {
            MyClass mc = new MyClass();

            Telescope.TelescopeNode node = new Telescope.TelescopeNode(mc, "My node", "mc");

            Console.ReadLine();
        }
    }

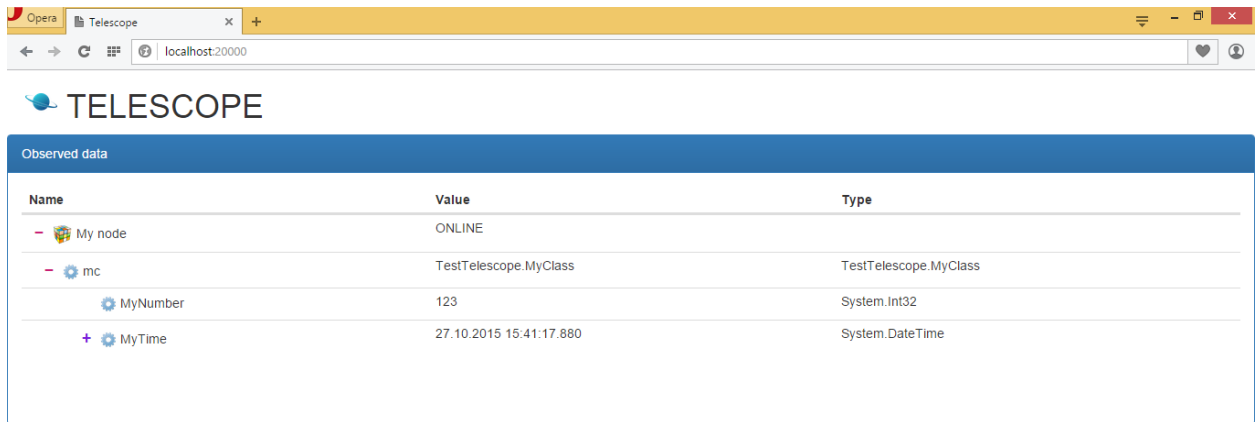
    public class MyClass
    {
        public MyClass()...

        public int MyNumber { get; set; }
        public DateTime MyTime { get { return DateTime.Now; } }
    }
}

```

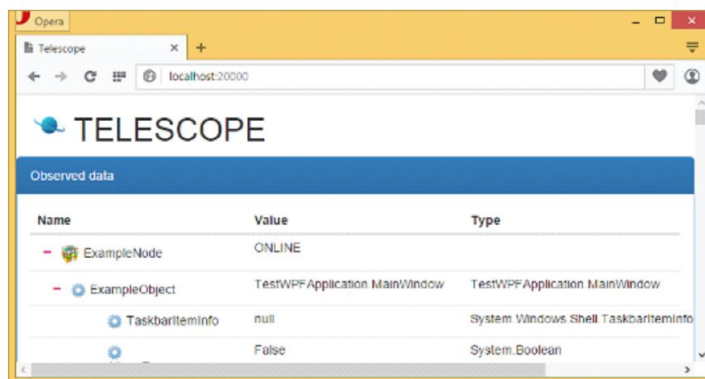
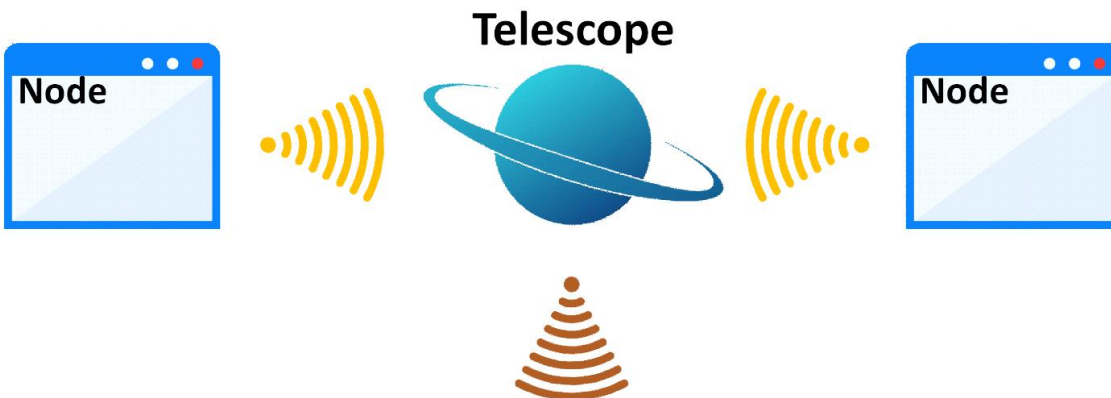
- Compile the project
- (Optional) Create configuration file for node `TelescopeNodeConfig.cfg` located in `<TargetDir>\config`. This file contains information about the **Telescope** port number. An example of such a file can be found in `<Telescope installation folder>\config`.
- Run **Telescope.exe** (run as Administrator)

8. Run your application (run as Administrator)
9. Enter your browser and go to http://localhost:<Telescope_port_number>. Using default settings it should read: <http://localhost:20000/>
10. Browser displays information about your objects. For example:

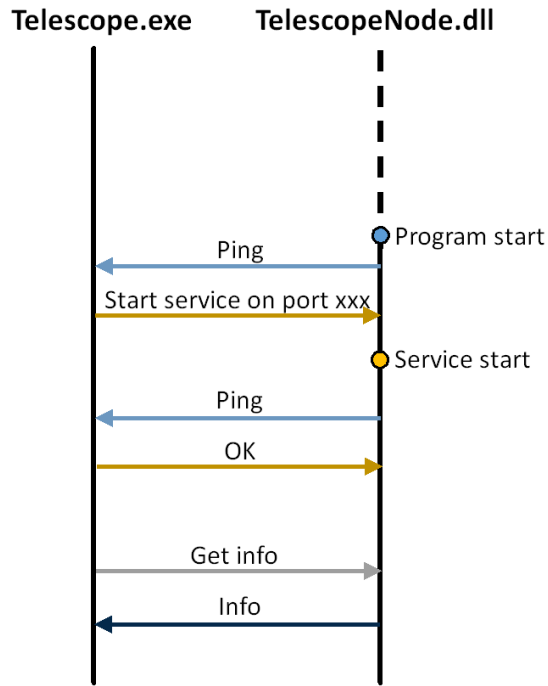


VI. How it works

The Telescope program collects information it gets from nodes located in your applications. When you make a request from the browser it sends requests to nodes. These gain information about your objects using System.Reflection namespace and send it back to **Telescope**. **Telescope** then gathers it together and wraps it in html response for the browser. These processes are depicted on the following scheme:



When you start your application with **TelescopeNode.dll** embedded in it, the node is not active; it cannot send messages until it connects **Telescope**. After connection, **Telescope** passes the port number to the node and then starts the service using this number. After that the node does become active and is able to obtain information about the object contained in the memory of your application. The interaction between node (**TelescopeNode.dll**) and aggregator (**Telescope.exe**) can be summarized as:



VII. Performance

TelescopeNode.dll receives data in a separate thread and does not use the cpu resources of your application. Telescope requires certain amount of memory for the temporary storage of the messages that are transmitted between TelescopeNode and Telescope. We plan to decrease this memory usage in future versions of our solution.