Introduction to PixEdit Server's web service interface

In this article we will show how to connect to the PixEdit Server Core and configure some job tickets to demonstrate different types of document processing.

PixEdit Server and PixEdit Converter Server hosts a web service interface to allow clients to connect and submit documents for processing and conversion. The interface provides a set of methods to allow clients to create, submit and monitor document processing jobs. Document processing could for instance be OCR, de skew, crop, resize, barcode separation or a simple file format conversion

In fact, the web service interface is used by the file pickup service 'PixEdit Input Connector' and the 'PixEdit Server Manager'.

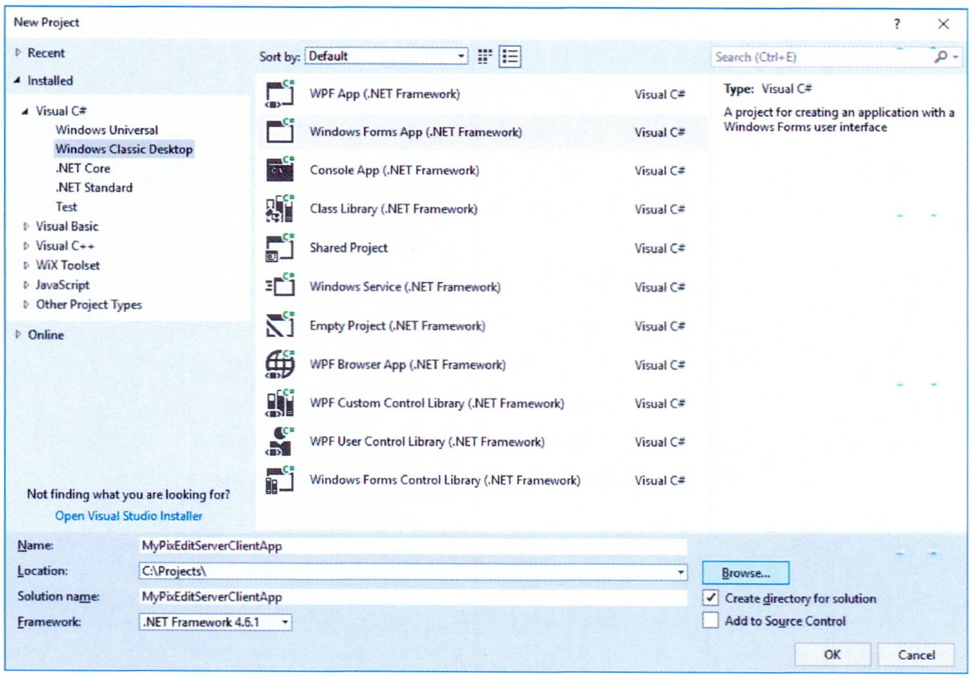
Before you start

The article requires basic knowledge of programming in C# with Microsoft Visual Studio. Here, we will use Visual Studio 2017, but you should be able to follow along with other versions also. Still, we recommend using VS 2013 or newer.

You should also have the newest version of PixEdit® Server installed in your development environment.

Set up your application

Open up Visual Studio and choose File, New, Project. In the left pane of the New Project dialog, browse to Visual C#, Windows and select it. Select Windows Forms Application in the right pane, choose a name and location for your project and click OK. (In this article, we will call our application MyPixEditServerClientApp.)


New Project
Recent
Installed
Visual C#
Windows Universal
Windows Classic Desktop
.NET Core
.NET Standard
Test
Visual Basic
Visual C++
WiX Toolset
JavaScript
Other Project Types
Online
Not finding what you are looking for?
Open Visual Studio Installer
Sort by: Default
Search (Ctrl+E)
WPF App (.NET Framework) Visual C#
Windows Forms App (.NET Framework) Visual C#
Console App (.NET Framework) Visual C#
Class Library (.NET Framework) Visual C#
Shared Project Visual C#
Windows Service (.NET Framework) Visual C#
Empty Project (.NET Framework) Visual C#
WPF Browser App (.NET Framework) Visual C#
WPF Custom Control Library (.NET Framework) Visual C#
WPF User Control Library (.NET Framework) Visual C#
Windows Forms Control Library (.NET Framework) Visual C#
Browse...
OK
Cancel

Type: Visual C#

A project for creating an application with a Windows Forms user interface

Name: MyPixEditServerClientApp

Location: C:\Projects\

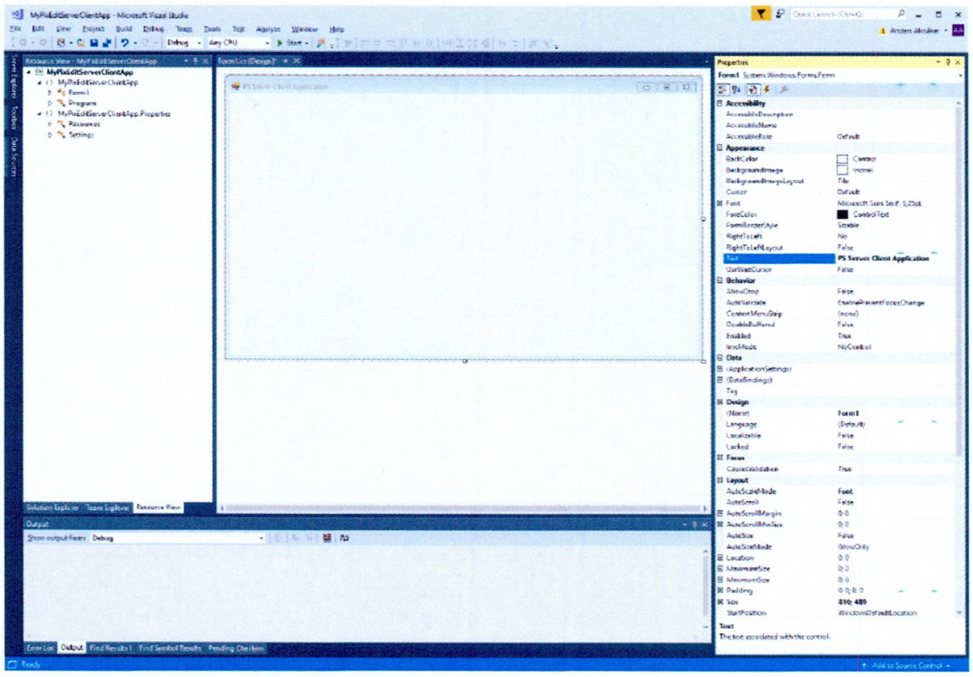
Solution name: MyPixEditServerClientApp

Framework: .NET Framework 4.6.1

[✓] Create directory for solution

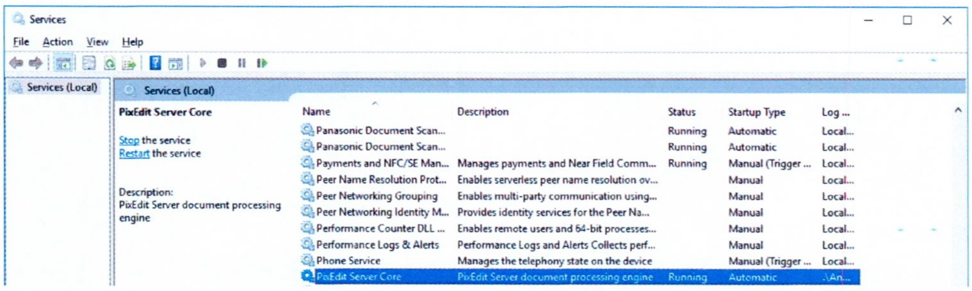
[ ] Add to Source Control

Visual Studio prepares your application and eventually, a basic, empty form called Form1 comes up. We rename the form to PSClientApp and set its title to PixEdit Server Client Application.


MyPixEditServerClientApp - Microsoft Visual Studio
Resource View - MyPixEditServerClientApp
Form1.cs [Design]
Properties
PS Server Client Application
Output

Connect to PixEdit Server

First make sure PixEdit Server is running preferable on your development computer.



PixEdit Server Core

Stop the service
Restart the service

Description:
PixEdit Server document processing engine

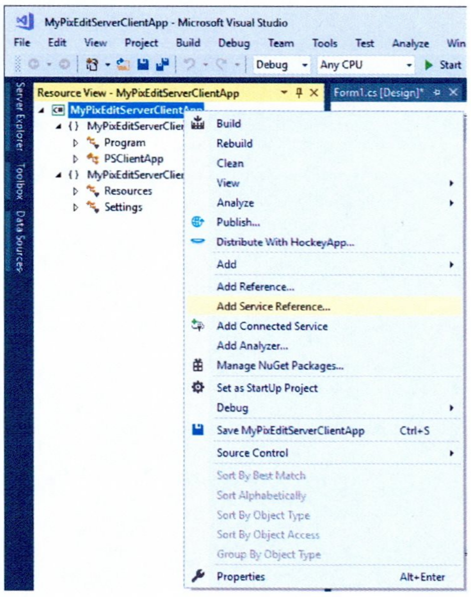
Name	Description	Status	Startup Type	Log ...
Panasonic Document Scan...		Running	Automatic	Local...
Panasonic Document Scan...		Running	Automatic	Local...
Payments and NFC/SE Man...	Manages payments and Near Field Comm...	Running	Manual (Trigger ...	Local...
Peer Name Resolution Prot...	Enables serverless peer name resolution ov...		Manual	Local...
Peer Networking Grouping	Enables multi-party communication using...		Manual	Local...
Peer Networking Identity M...	Provides identity services for the Peer Na...		Manual	Local...
Performance Counter DLL ...	Enables remote users and 64-bit processes...		Manual	Local...
Performance Logs & Alerts	Performance Logs and Alerts Collects perf...		Manual	Local...
Phone Service	Manages the telephony state on the device		Manual (Trigger ...	Local...
PixEdit Server Core	PixEdit Server document processing engine	Running	Automatic	.\An...

To access the web service interface, we will need to know its endpoint or URL address. The PixEdit server Core writes this to the Windows Event Viewer when it starts. Like this:

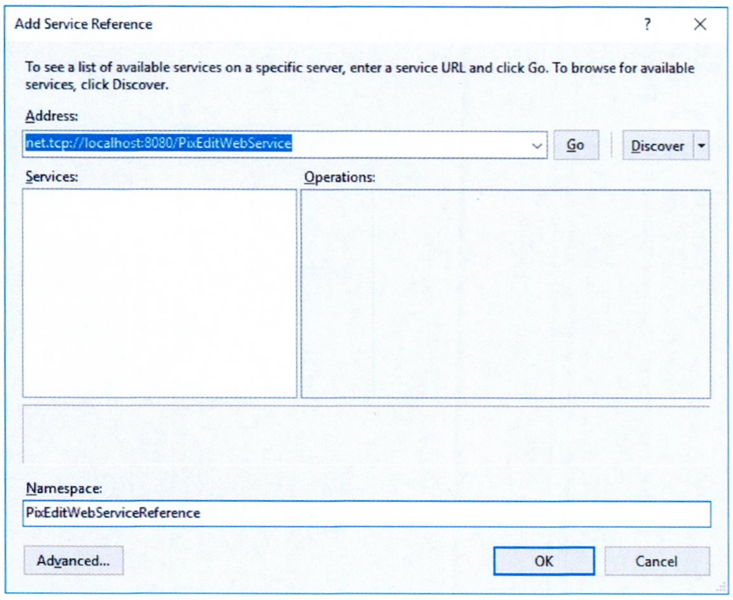
```
Listening for messages at:
net.tcp://tsws19:8080/PixEditWebService (NetTcpBinding)
net.tcp://localhost:8080/PixEditWebService/mex (MetadataExchangeTcpBinding)
http://tsws19:8180/PixEditWebService (WSHttpBinding)
urn:docs-oasis-open-org:ws-dd:ns:discovery:2009:01 (CustomBinding)
```

PixEdit provides two possible bindings. Net.tcp which is a duplex binding and will work inside your intranet. The http binding is simplex and will work for server/clients outside your intranet. In this tutorial we will go with the net.tcp binding which will allow us to implement some callback features creating a very tight and responsive client application.

Now head back to Visual studio. In the solution explorer right click and select 'Add Service reference'



In this dialog paste the endpoint address and press the 'Go' button to discover the PixEditWebService


Add Service Reference

To see a list of available services on a specific server, enter a service URL and click Go. To browse for available services, click Discover.

Address:
net.tcp://localhost:8080/PixEditWebService

Go | Discover

Services: | Operations:

Namespace:
PixEditWebServiceReference

Advanced... | OK | Cancel

Then rename the Namespace to 'PixEditWebServiceReference' and press ok to add the service reference to the project.

Adding basic code to get things up and running

With the web service up and running its time to add some code using it. In the solution explorer right click the form1.cs and select view code.

First, we need to add a reference to the web service

```
using MyPixEditServerClientApp.PixEditWebServiceReference;
```

Since we are connecting using the net.tcp binding we need to implement the callback features from the web service. We need to derive from the `IPixEditWebServiceCallback` and set the callback attribute, like this:

```
[CallbackBehavior(UseSynchronizationContext = false)]
public partial class PSClientApp : Form, IPixEditWebServiceCallback
```

Now above the form load handler add an instance of our PixEdit web Service and a GUID to hold a user id. In the form load handler add the following code to connect to the service and validate a user. PixEdit Server has its own user system to identify which processing jobs each user has submitted for processing. The user id is needed for about every method in this tutorial but we will not get into any details of the user management besides that. The 'admin' user is a default user created by the core service on the initial startup.

```
PixEditWebServiceClient _pixClient;
Guid _clientUserID;

private void PSClientApp_Load(object sender, EventArgs e)
{
            InstanceContext context = new InstanceContext(this);
            _pixClient = new PixEditWebServiceClient(context,
"NetTcpBinding_IPixEditWebService", "net.tcp://localhost:8080/PixEditWebService");
            _clientUserID = _pixClient.ValidateUser("admin", "admin");

}
```

We also need to add the 3 callback functions the interface provide or the project will not compile.

```
public void OnServerStatus(string status)
{
}
public void OnJobStatusChanged(string status)
{
}
public void OnJobProfilesChanged()
{
}
```

Ready to do some document processing

Our application should now compile and run okay but it doesn't do very much besides from connecting and validating the default user.

To process documents, you will need to set up a job ticket. The ticket specifies the different kind of imaging operations for the PixEdit core to apply to the document.

In this sample we would like to apply the following operations:

Remove blank pages Usually needed when scanning both front and backside of your paper documents

Remove black borders Removes black borders produced by most high-end document scanners. This function also corrects for any skew that may arise during the scan process

Erase borders This will delete any remaining noise near the document edges

Resize to standard page sizes This function will snap each page in the document to exact paper sizes, like for instance A4 and A3. This could be required if the processed document is being handled by other services in the system.

OCR Recognize text to make our processed documents searchable.

Save properties Specify saving properties like naming convention, compression and type of file format

There is no supplied specification for the job ticket xml. But the PixEdit Server Manager tool uses the same file format when users design job profiles there. So, one can simple use the tool to investigate the different processing options and later have a look inside the Jobprofiles.xml located in the product's working folder c:\DocServer

The xml job ticket for our job will look like this: