JetTrac Connect - creating your own custom module

The ***JetTrac JobController*** is an incredibly powerful processing engine with a very open architecture. It is an integral part of the ***JetTrac Connect*** Platform. Over the past 25 years we have designed the ***JetTrac JobController*** to allow our customers to create their own custom modules and to insert them into a ***JetTrac*** job, running as a step of an overall production job and combining them with standard ***JetTrac*** modules. The purpose of this document is to give an overview of how a software developer can develop a custom module that will run as part of a job in the ***JetTrac JobController***.

First, the WHY:
The ***JetTrac Modules*** cover a wide variety of processing logic, however, you may have a requirement to develop some custom logic/processing that is not covered by an existing ***JetTrac Module***.

Examples might be:

- "Connect" disparate systems that don't talk to each other. The ***JetTrac platform*** could likely connect these systems with the addition of a custom module specific to your requirements and environment, that would perform the necessary functions to make the connection (we call this "Tight Integration").
- A tight integration to a workflow or content management system with a production ***JetTrac*** process.
- Custom integration with a specific database to enable that database to be updated at a specific step of a ***JetTrac*** job.
- A tight integration with an accounting system, e.g. when a work order is approved, using an SDK that will programmatically send invoice data in a specific format to your system to create an invoice.
- Develop some custom logic with the data that is being processed through ***JetTrac Connect.***

Now, the HOW:
There are several parts to developing a functional ***JetTrac*** custom module:

1. Appoint a software developer who can write a program in whatever programming language that you wish to use, as long as the program can be compiled into some kind of executable (or script) that can be called using a command line in a Windows environment. All of ProTechnology's development in the last 20 years has been either C#/.NET development or Java development however Ruby, Python, JavaScript, Perl, Visual Basic or any other programming language can be used that meets the criteria below.

2. The program must accept a set of data/parameters on the command line (more on this below).

3. The program must be able to return a status code of 0 to the calling program (which is the ***JetTrac JobController***). If the program ran unsuccessfully, a 0 is displayed to indicate a problem.

4. Ideally, the program should write to a log file in a format similar to below:

   20201209072454 JobController : info Starting agent: C:\Perl64\bin\perl.exe "C:\JetTrac\Programs\JTDataPrep_XML\JTDataPrep_XML.pl" "C:\JetTrac\Forms\JetTrac_QuickBooks_Invoice\Temp\JTDataPrep_Output.xml" "C:\JetTrac\Forms\JetTrac_QuickBooks_Invoice\Temp\JTDataPrep_XML_Output.xml" "C:\JetTrac\Forms\JetTrac_QuickBooks_Invoice\Config\JTDataPrep_XML_Config.ini" "C:\JetTrac\JobController\Server\Log\JobController.log"
   202012972454 JTDataPrep_XML (v1.3, 5/14/2019): Replaced < with < in tag <ToEmail>.
   202012972454 JTDataPrep_XML (v1.3, 5/14/2019): Replaced > with > in tag <ToEmail>.
   202012972454 JTDataPrep_XML (v1.3, 5/14/2019): Program executed successfully.
   20201209072455 JobController : info Agent execution completed normally.

The Command Line and Command Line Parameters

It is absolutely essential that your custom module is able to run on a command line in a Windows environment. The program will know what to do based on information passed to it on the command line. The common parameter types used are the following:

- Input filename (XML, PDF, etc.)
- Output filename
- Configuration/ini filename (more on that below)
- Log filename

Here is a specific example of one of our standard ***JetTrac Modules*** called JetTrac XMLTrans. The purpose of JetTrac XMLTrans is to read in an XML file, do some manipulations to the XML data and write out a modified XML file that normally has some additional information in it added by JetTrac XMLTrans. Here is a standard command line:

JTXMLTrans.exe Input.xml Output.xml JTXMLTrans.ini JTXMLTrans.log

Let's break down each part of the above line:

1. JTXMLTrans.exe - this is the name of the executable program, normally fully qualified to the exact location where the executable is located

2. Input.xml - the name of the input XML that contains some data used in the program logic that may be used to define other data to be written to the output file

3. Output.xml - the name of the output XML that contains the result of running the program

4. JTXMLTrans.ini - this is a configuration file that is read by JTXMLTrans that tells the program what to do - more on this below

5. JTXMLTrans.log - the name of the log file to write log info

Let's look at a very simple example of an input XML file named Input.xml:

```
<?xml version="1.0"?>
<Document>
	<CustomerFirstName>Fred</CustomerFirstName>
	<CustomerLastName>Flintstone</CustomerLastName>
	<CustomerID>123456</CustomerID>
	<Date>20161020</Date>
</Document>
```

One of the many functions that has been programmed into JTXMLTrans is the ability to create a new field by concatenating other fields and strings of data. Let's say there was a requirement to put a field on a form in the format of First Name, a space, Last Name, a space then a dash then a space then the word ID then a space then the value of CustomerID. This is how you would configure this concatenation in a configuration file named JTXMLTrans.ini :

```
<?xml version="1.0"?>
<Config>
<Configure_JetTrac_XMLTrans>
	<CreateField>
		<FieldName>NameAndID</FieldName>
		<Term>CustomerFirstName</Term>
		<Term>" "</Term>
		<Term>CustomerLastName</Term>
		<Term>" - ID "</Term>
		<Term>CustomerID</Term>
	</CreateField>
</Configure_JetTrac_XMLTrans>
</Config>
```

Now that we have an Input.xml and a configuration file we can now test from a bat file (if you have access to this executable). Let's create the file named "Run JTXMLTrans.bat" with the following content:

```
JTXMLTrans.exe   Input.xml   Output.xml   JTXMLTrans.ini   JTXMLTrans.log
Pause
```

You should get a new file created named Output.xml with this content:

```
<?xml version="1.0"?>
<Document>
        <CustomerFirstName>Fred</CustomerFirstName>
        <CustomerLastName>Flintstone</CustomerLastName>
        <CustomerID>123456</CustomerID>
        <Date>20161020</Date>
        <NameAndID>Fred Flintstone - 123456</NameAndID>
</Document>
```

Hopefully this simple example will give you an idea why you may want or need to create a custom module that would run within ***JetTrac Connect*** and how you can set it up. The logic in the custom modules can be as complex as you want it to be, however HOW it runs within ***JetTrac JobController*** is very straightforward.

Registering a custom module to appear in the module list in ***JetTrac JobConfig*** is simple once you understand the structure of the ***JetTrac*** configuration file. There is a separate document providing those details.

Contact Matt Sliwa at matthew.sliwa@protechinc.com if you would like more information on how to create a ***JetTrac Custom Module***.

Source document: Google Drive: JetTrac:JetTrac Modules:JetTrac Create Your Own Custom Module