Cambridge Technology XML API Quick Start Guide

1 Purpose

The purpose of this document is to introduce the Cambridge Technology XML API. It will highlight where it can be used. For specific XML commands, see the Software Reference Manual for available XML command definitions.

2 Introduction

The Cambridge Technology XML API is an API that allows you to use XML to develop scan jobs for the ScanMaster Controller (SMC).

NOTE: Before getting started with XML API, make sure the hardware is operational and all necessary software is installed. See prerequisites below.

The ScanMaster API (SMAPI) provides an API that allows for the development of a custom Microsoft Windows user application. It currently supports more features than the XML API. For more information about the SMAPI, see the ScanMaster API Reference Manual.

Scan jobs (XML API code), written in XML, can be run from the MiniEditor or from code within a Visual Studio host application. Scan jobs also can be run from MiniEditor without a Microsoft Windows host application. Additionally, they can be downloaded to the SMC and run without a human interface, if desired.

The following figure shows how the ScanMaster Controller interacts with both the XML API and SMAPI.

2.1 About the XML API

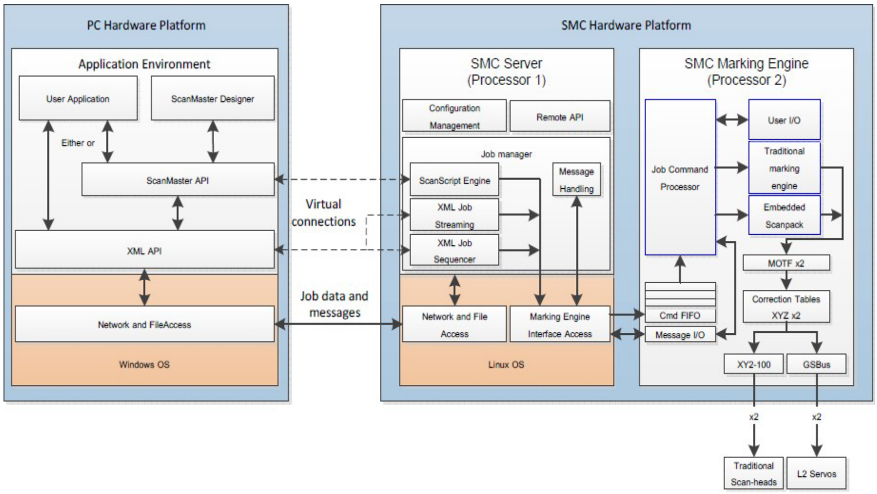


Figure 1 - **XML and SMAPI**

3 Prerequisites

Before getting started with XML API, there are a few prerequisites to deal with:

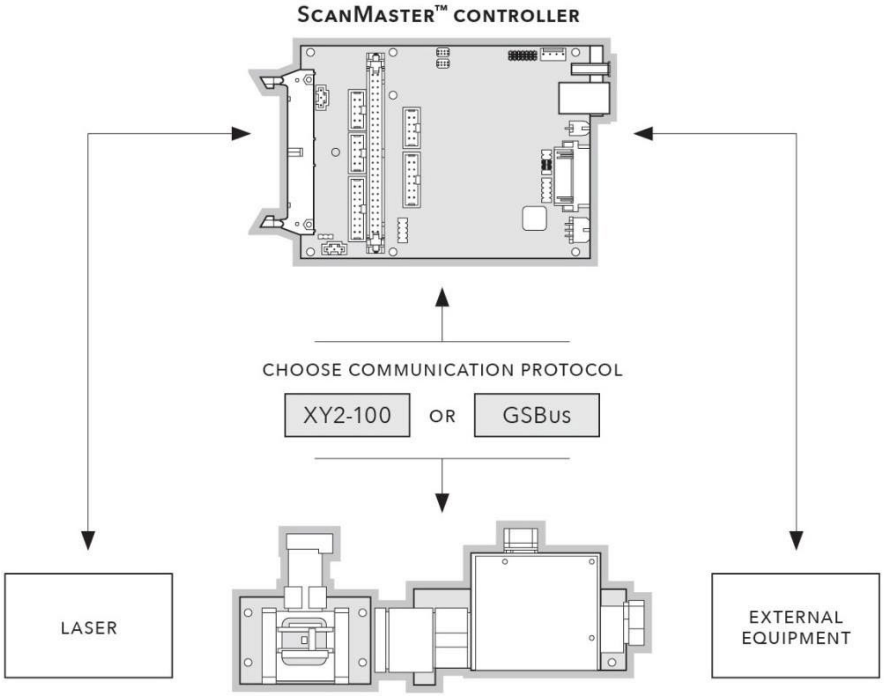


Figure 2 - **Hardware – SMC, laser, scan head**

Be sure your ScanMaster Controller, laser and scan head are operational.

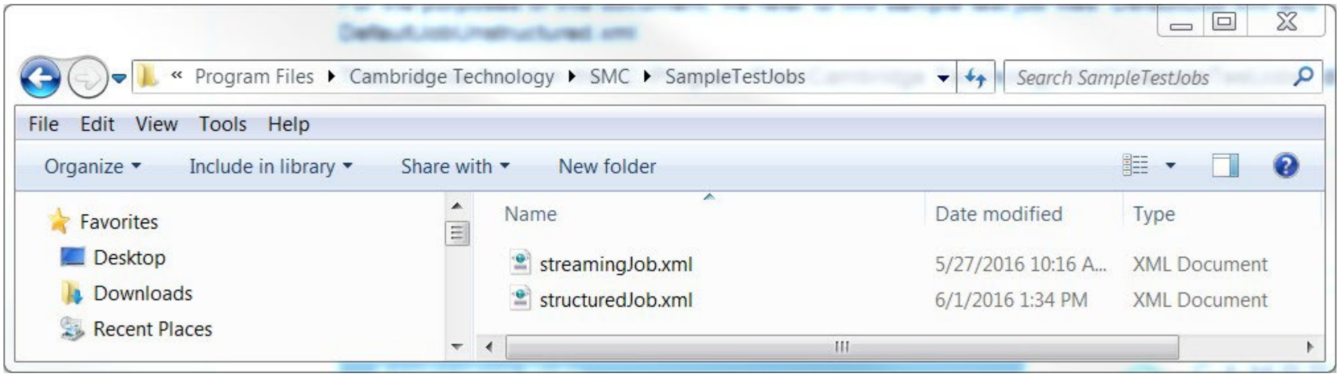
1. If you plan to have a custom Microsoft Windows application host your XML programs, make sure Microsoft Visual Studio (Version 10 or greater) is installed (see the Microsoft website).
2. Install the most up-to-date SMAPI and the ScanMaster Controller SDK (see Cambridge Technology Software Suite Setup for details about these software packages).
3. Install the Cambridge Technology MiniEditor from: http://www.camtech.com/downloads/customers/Software/ec1000minieditor_v450.zip

NOTE: If required, contact your Cambridge Technology representative for the Download page password.

4 Example Scan Jobs

For the purposes of this document, we refer to two sample test job files: streamingJob.xml and structuredJob.xml

They can be found in the “C:\Program Files\Cambridge Technology\SMC\SampleTestJobs” directory.



Name	Date modified	Type
streamingJob.xml	5/27/2016 10:16 A...	XML Document
structuredJob.xml	6/1/2016 1:34 PM	XML Document

Figure 3 - **Sample test job directory**

5 A Typical XML Structured Scan Job (structuredJob.xml)

Scan jobs consist of XML command statements. XML scan jobs can be structured into groups of related statements called segments, and these segments can be sent to the SMC as named entities for deferred execution. This structure allows these named segments to be reused by name in a sequence. You can also create unstructured jobs without named segments. The XML commands in an unstructured scan job are only run once in the run of the scan job.

NOTE: This section refers to the structuredJob.xml example (see Figure 3 above).

The following is an XML named code segment that scans a rectangle:

```
<Segment id='Box' iterations='1' deferred='TRUE'>
<!-- Defines the geometry of a box -->
<!-- Be sure geometry is defined after laser parameters are set. -->
<JumpAbs>-25000,-5000</JumpAbs>
<MarkAbs>25000,-5000</MarkAbs>
<MarkAbs>25000,5000</MarkAbs>
<MarkAbs>-25000,5000</MarkAbs>
<MarkAbs>-25000,-5000</MarkAbs>
</Segment>
```

Deferred segments are executed later, during the sequence execution from a RunSegment command. The RunSegment command is executed from inside a sequence.

```
<Sequence iterations='1'>
<!-- Main sequence of operation -->
<RunSegment>Preamble</RunSegment>
<RunSegment>Alignment</RunSegment>
<RunSegment>Params:Default</RunSegment>
<RunSegment>Box</RunSegment>
<RunSegment>Postamble</RunSegment>
</Sequence>
```

The following sections describe the recommended segments and their use for a typical scan job.

5.1 Structured Job Segments: Preamble

In most structured jobs there are configuration commands that need to happen before the main part of the scan job happens. The preamble is where commands like “<set id='EnableLaser'>TRUE</set>” belong.

The preamble from the sample enables the laser:

```
<Segment id='Preamble' deferred='True'>
<!-- Initial configuration as necessary -->

<!-- Begin required first -->
<BeginJob></BeginJob>
      <!-- Position units in marking field will be in 16 bit format -->
<!-- for this example.  Most systems will be calibrated in more -->
<!-- familiar units such as millimeters, and may use more -->
<!-- resolution if the hardware in the system supports it -->
<set id='ActuatorUnits'>bits-16</set>
<!-- Scale set to one prior to calibration -->
<set id='XYCalFactor'>1</set>
<set id='ZCalFactor'>1</set>
<!-- Turns off guide laser as necessary and readies main laser -->
<set id='EnableLaser'>TRUE</set>
</Segment>
```

5.2 Structured Job Segments: Alignment

The alignment segment allows for the adjustment of the image field offset, scaling and rotation. These adjustments are usually done before any segments that scan and are typically in effect for the run of the entire scan job or until redefined.

```
<Segment id='Alignment' deferred='True'>
<!-- This segment allows for individual alignment corrections -->
<!-- Values here result in no change -->
<set id='FieldOffset'>0.000000; 0.000000; 0.000000</set>
<set id='Transform'>1.000000; 0.000000; 0.000000; 1.000000</set>
</Segment>
```

5.3 Structured Job Segments: Params

There are often parameters that need to be set that affect the laser or galvo movement settings. They are typically set before any scanning is done.

```
<Segment id='Params:Default' deferred='TRUE'>
<!-- This segment defines the desired behavior of the laser -->
<!-- A simple CO2 laser configuration with 20KHz modulation -->
<!-- at a nominal 20% duty cycle is shown here-->

<!-- This section overrides values stored in the configuration file -->
<!-- Setting timing resolution: 50 * 20ns base = 1us "tick" -->
<set id='LaserTiming'>50</set>
 <!-- CO2 does not require a delay -->
<set id='LaserEnableDelay'>0</set>
<!-- Normally zero except for some digital servos -->
<set id='LaserPipelineDelay'>0</set>
<!-- First Pulse Killer not required -->
<set id='LaserFPK'>0,0</set>
<!-- No modulation delay required -->
<set id='LaserModDelay'>0</set>
 <!-- CO2 does not require a timeout -->
<set id='LaserEnableTimeout'>0</set>
<!-- "Tickle pulse", Laser 2, 2 ticks wide, 100 tick period -->
<set id='LaserStandby'>2,10,100</set>
<!-- "Tickle pulse", Laser 1, 2 ticks wide, 100 tick period -->
<set id='LaserStandby'>1,1,100</set>

<!-- The following settings are essential for every job -->
<!-- CO2 does not require a delay -->
<set id='LaserOnDelay'>0</set>
<!-- deassert LASER_GATE 50 ticks after microvectoring complete -->
<set id='LaserOffDelay'>50</set>
<!-- 20kHz, 20% duty cycle -->
<!-- Operating pulse, Laser1, 10 ticks wide, 50 tick period -->
<set id='LaserPulse'>1,10,50</set>
<!-- Laser 2 signal, here set identical to laser 1 -->
<set id='LaserPulse'>2,5,10</set>
<!-- Mark speed in us per micro-step, and bits per microstep-->
<!-- in this example, a full field 65536 bits wide would take -->
<!-- 65536/2 * 10 us or 327.68 ms -->
<Set id='MarkSpeed'>10,2</Set>
<!-- Jump speed expressed similar to mark speed -->
<Set id='JumpSpeed'>10,8</Set>
</Segment>
```

5.4 Structured Job Segments: Postamble

In most structured jobs there are configuration commands that need to happen after the main part of the scan job happens. The postamble is where commands like “<set id='EnableLaser'>FALSE</set>” belong.

```
<Segment id='Postamble' deferred='True'>
<!-- In this segment, do all required cleanup -->
<set id='EnableLaser'>FALSE</set>
<EndJob></EndJob>
</Segment>
```

The full DefaultJob.xml is shown below:

```
<Data type='JobData' rev='2.0'>

<Segment id='Preamble' deferred='True'>
<!-- Initial configuration as necessary -->

<!-- Begin required first -->
<BeginJob></BeginJob>
        <!-- Position units in marking field will be in 16 bit format -->
<!-- for this example.  Most systems will be calibrated in more -->
<!-- familiar units such as millimeters, and may use more -->
<!-- resolution if the hardware in the system supports it -->
<set id='ActuatorUnits'>bits-16</set>
<!-- Scale set to one prior to calibration -->
<set id='XYCalFactor'>1</set>
<set id='ZCalFactor'>1</set>
<!-- Turns off guide laser as necessary and readies main laser -->
<set id='EnableLaser'>TRUE</set>
</Segment>

<Segment id='Alignment' deferred='True'>
<!-- This segment allows for individual alignment corrections -->
<!-- Values here result in no change -->
<set id='FieldOffset'>0.000000; 0.000000; 0.000000</set>
<set id='Transform'>1.000000; 0.000000; 0.000000; 1.000000</set>
</Segment>

<Segment id='Params:Default' deferred='TRUE'>
<!-- This segment defines the desired behavior of the laser -->
<!-- A simple CO2 laser configuration with 20KHz modulation -->
<!-- at a nominal 20% duty cycle is shown here-->

<!-- This section overrides values stored in the configuration file -->
```

```
<!-- Setting timing resolution: 50 x 20ns base = 1us "tick" -->
<set id='LaserTiming'>50</set>
 <!-- CO2 does not require a delay -->
<set id='LaserEnableDelay'>0</set>
<!-- Normally zero except for some digital servos -->
<set id='LaserPipelineDelay'>0</set>
<!-- First Pulse Killer not required -->
<set id='LaserFPK'>0,0</set>
<!-- No modulation delay required -->
<set id='LaserModDelay'>0</set>
 <!-- CO2 does not require a timeout -->
<set id='LaserEnableTimeout'>0</set>
<!-- "Tickle pulse", Laser 2, 2 ticks wide, 100 tick period -->
<set id='LaserStandby'>2,10,100</set>
<!-- "Tickle pulse", Laser 1, 2 ticks wide, 100 tick period -->
<set id='LaserStandby'>1,1,100</set>

<!-- The following settings are essential for every job -->
<!-- CO2 does not require a delay -->
<set id='LaserOnDelay'>0</set>
<!-- deassert LASER_GATE 50 ticks after microvectoring complete -->
<set id='LaserOffDelay'>50</set>
<!-- 20kHz, 20% duty cycle -->
<!-- Operating pulse, Laser1, 10 ticks wide, 50 tick period -->
<set id='LaserPulse'>1,10,50</set>
<!-- Laser 2 signal, here set identical to laser 1 -->
<set id='LaserPulse'>2,5,10</set>
<!-- Mark speed in us per micro-step, and bits per microstep-->
<!-- in this example, a full field 65536 bits wide would take -->
<!-- 327.68 ms -->
<Set id='MarkSpeed'>10,2</Set>
<!-- Jump speed expressed similar to mark speed -->
<Set id='JumpSpeed'>10,8</Set>
</Segment>

<Segment id='Box' iterations='1' deferred='TRUE'>
<!-- Defines the geometry of a box -->
<!-- Be sure geometry is defined after laser parameters are set. -->
<JumpAbs>-25000,-5000</JumpAbs>
<MarkAbs>25000,-5000</MarkAbs>
<MarkAbs>25000,5000</MarkAbs>
<MarkAbs>-25000,5000</MarkAbs>
```

```
<MarkAbs>-25000,-5000</MarkAbs>
</Segment>

<Segment id='Postamble' deferred='True'>
<!-- In this segment, do all required cleanup -->
<set id='EnableLaser'>FALSE</set>
<EndJob></EndJob>
</Segment>

<Sequence iterations='1'>
<!-- Main sequence of operation -->
<RunSegment>Preamble</RunSegment>
<RunSegment>Alignment</RunSegment>
<RunSegment>Params:Default</RunSegment>
<RunSegment>Box</RunSegment>
<RunSegment>Postamble</RunSegment>
</Sequence>
</Data>
```

6 An Unstructured Scan Job (streamingJob.xml)

While it is recommended that you use structured scan jobs in most situations, you can create unstructured scan jobs as well. The unstructured version of the above structured job is shown below.

NOTE: This section refers to the streamingJob.xml example (see Figure 3 above).

The full unstructured scan job is shown below:

```
<Data type='JobData' rev='2.0'>

<!-- Initial configuration as necessary -->
<!-- Begin required first -->
<BeginJob></BeginJob>
      <!-- Position units in marking field will be in 16 bit format -->
 <!-- for this example.  Most systems will be calibrated in more -->
<!-- familiar units such as millimeters, and may use more -->
<!-- resolution if the hardware in the system supports it -->
<set id='ActuatorUnits'>bits-16</set>
<!-- Scale set to one prior to calibration -->
<set id='XYCalFactor'>1</set>
<set id='ZCalFactor'>1</set>
<!-- Turns off guide laser as necessary and readies main laser -->
<set id='EnableLaser'>TRUE</set>
```

```
<!-- Allows for individual alignment corrections -->
<!-- Values here result in no change -->
<set id='FieldOffset'>0.000000; 0.000000; 0.000000</set>
<set id='Transform'>1.000000; 0.000000; 0.000000; 1.000000</set>

<!-- Defines the desired behavior of the laser -->
<!-- A simple CO2 laser configuration with 20KHz modulation -->
<!-- at a nominal 20% duty cycle is shown here-->

<!-- This section overrides values stored in the configuration file -->
<!-- Setting timing resolution: 50 * 20ns base = 1us "tick" -->
<set id='LaserTiming'>50</set>
 <!-- CO2 does not require a delay -->
<set id='LaserEnableDelay'>0</set>
<!-- Normally zero except for some digital servos -->
<set id='LaserPipelineDelay'>0</set>
<!-- First Pulse Killer not required -->
<set id='LaserFPK'>0,0</set>
<!-- No modulation delay required -->
<set id='LaserModDelay'>0</set>
 <!-- CO2 does not require a timeout -->
<set id='LaserEnableTimeout'>0</set>
<!-- "Tickle pulse", Laser 2, 2 ticks wide, 100 tick period -->
<set id='LaserStandby'>2,10,100</set>
<!-- "Tickle pulse", Laser 1, 2 ticks wide, 100 tick period -->
<set id='LaserStandby'>1,1,100</set>

<!-- The following settings are essential for every job -->
<!-- CO2 does not require a delay -->
<set id='LaserOnDelay'>0</set>
<!-- deassert LASER_GATE 50 ticks after microvectoring complete -->
<set id='LaserOffDelay'>50</set>
<!-- 20kHz, 20% duty cycle -->
<!-- Operating pulse, Laser1, 10 ticks wide, 50 tick period -->
<set id='LaserPulse'>1,10,50</set>
<!-- Laser 2 signal, here set identical to laser 1 -->
<set id='LaserPulse'>2,5,10</set>
<!-- Mark speed in us per micro-step, and bits per microstep-->
<!-- in this example, a full field 65536 bits wide would take -->
<!-- 65536/2 * 10 us or 327.68 ms -->
<Set id='MarkSpeed'>10,2</Set>
<!-- Jump speed expressed similar to mark speed -->
```

```
<Set id='JumpSpeed'>10,8</Set>

<!-- Defines the geometry of a box -->
<!-- Be sure geometry is defined after laser parameters are set. -->
<JumpAbs>-25000,-5000</JumpAbs>
<MarkAbs>25000,-5000</MarkAbs>
<MarkAbs>25000,5000</MarkAbs>
<MarkAbs>-25000,5000</MarkAbs>
<MarkAbs>-25000,-5000</MarkAbs>

<!-- Do all required cleanup -->
<set id='EnableLaser'>FALSE</set>
<EndJob></EndJob>

</Data>
```

7 Creating a New XML Scan Job

A new XML scan job consists of an XML file with the following lines:

```
<Data type='JobData' rev='2.0'>        <!-- A Data type tag starts every XML job file       -->
<BeginJob />                           <!-- A BeginJob/ tag starts every job code section -->
                                       <!—Your XML job code lines here                    -->
<EndJob/>                              <!-- A EndJob/ tag ends every job code section     -->
</Data>                                <!-- A /Data tag ends every XML job file           -->
```

You can create a new XML job file with an XML editor or even Notepad as long as you have these lines. You then add your job code between the BeginJob and EndJob tags. Another option is to open (import into MiniEditor) a sample job like DefaultJob.xml and then change the code in between the BeginJob and EndJob tags. See the Software Reference manual for available XML command definitions.

8 Using the MiniEditor

The MiniEditor is a PC-based editor designed to allow you to create, edit and test XML scan jobs without the need of a Visual Studio host application.

It provides a number of useful XML job related features, including:

- Generating full XML job syntax code automatically.
- Running XML jobs without the use of Visual Studio.
- Toggling I/O signals on the board.
- Changing settings used when jobs run.

To open the MiniEditor from the start menu:

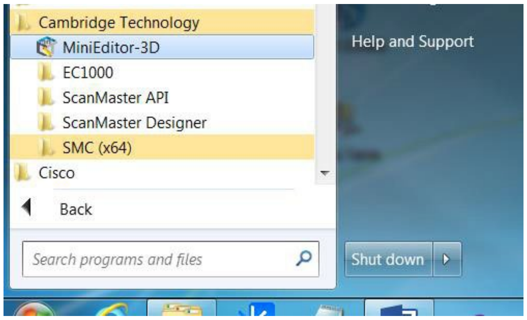


Figure 4 - **Select MiniEditor-3D from start menu**

1. Click MiniEditor-3D.exe to start it. The MiniEditor will open:

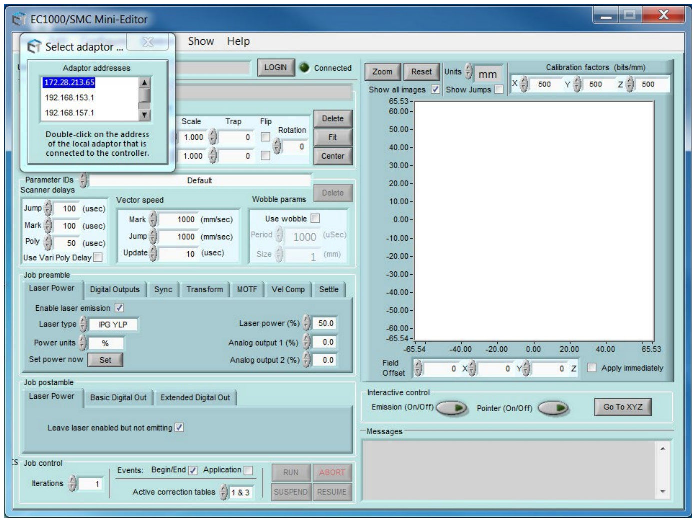
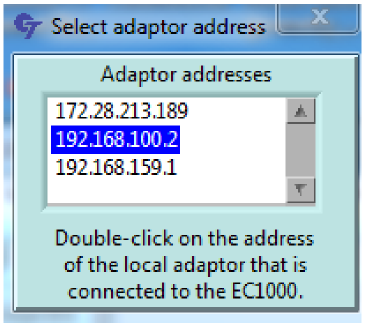


Figure 5 - **MiniEditor**

2. When the MiniEditor opens, select (double click) the IP address of the PC connected to the SM/EC1000 OR SMC controller.


Select adaptor address

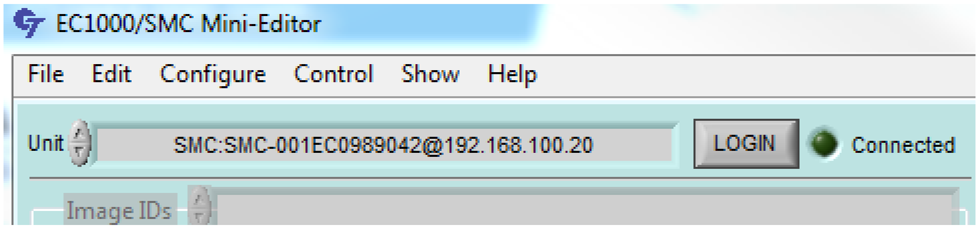
Adaptor addresses

172.28.213.189
192.168.100.2
192.168.159.1

Double-click on the address of the local adaptor that is connected to the EC1000.

Figure 6 - **Select (double click) your computer's IP address**

3. Select the controller unit:



Unit: SMC:SMC-001EC0989042@192.168.100.20

Figure 7 - **Select the controller**

4. Click **LOGIN**:

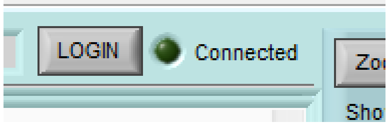
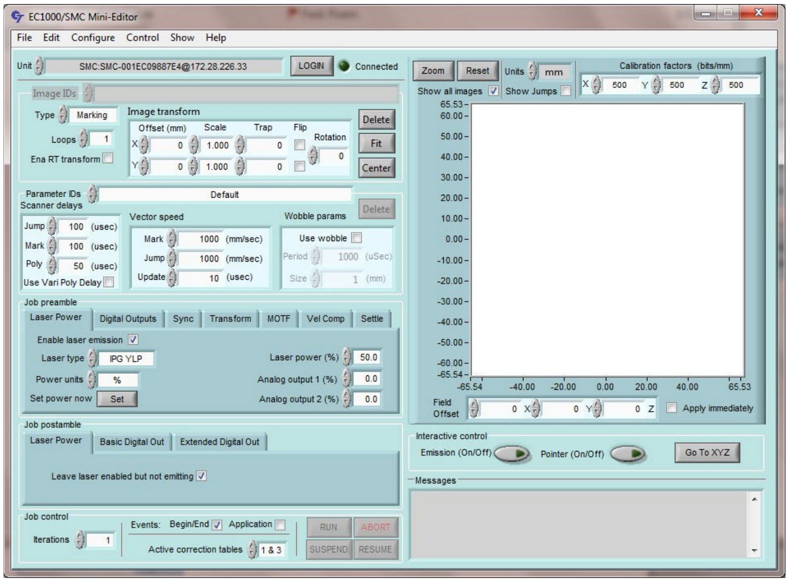


Figure 8 - **Login button**

Once MiniEditor is open, it looks like this:



Unit: SMC:SMC-001EC09887E4@172.28.226.33

Type: Marking
Loops: 1
Ena RT transform □

Offset (mm) X: 0, Y: 0 | Scale X: 1.000, Y: 1.000 | Trap X: 0, Y: 0 | Flip X: □, Y: □ | Rotation: 0

Parameter IDs: Default

Scanner delays: Jump 100 (usec), Mark 100 (usec), Poly 50 (usec), Use Vari Poly Delay □

Vector speed: Mark 1000 (mm/sec), Jump 1000 (mm/sec), Update 10 (usec)

Wobble params: Use wobble □, Period 1000 (uSec), Size 1 (mm)

Enable laser emission ☑
Laser type: IPG YLP
Power units: %
Set power now
Laser power (%): 50.0
Analog output 1 (%): 0.0
Analog output 2 (%): 0.0

Leave laser enabled but not emitting ☑

Iterations: 1
Events: Begin/End ☑ Application □
Active correction tables: 1 & 3

Units: mm
Show all images ☑
Show Jumps □
Calibration factors (bits/mm): X 500, Y 500, Z 500

Field Offset: 0 X, 0 Y, 0 Z
Apply immediately □

Emission (On/Off)
Pointer (On/Off)

Figure 9 - **MiniEditor**

8.1 Importing an XML Job

MiniEditor started as a debugging and testing tool and does not create an XML job file from scratch. You can create an empty text file in notepad and save it as a XML document. From the MiniEditor, you can open and edit existing XML scan jobs.

To open a saved XML job from your PC:

1. From the File menu, select **Import …XML job file**.

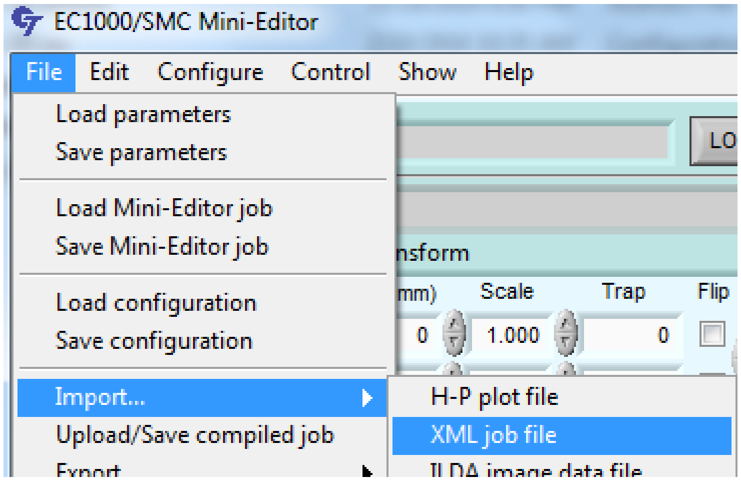
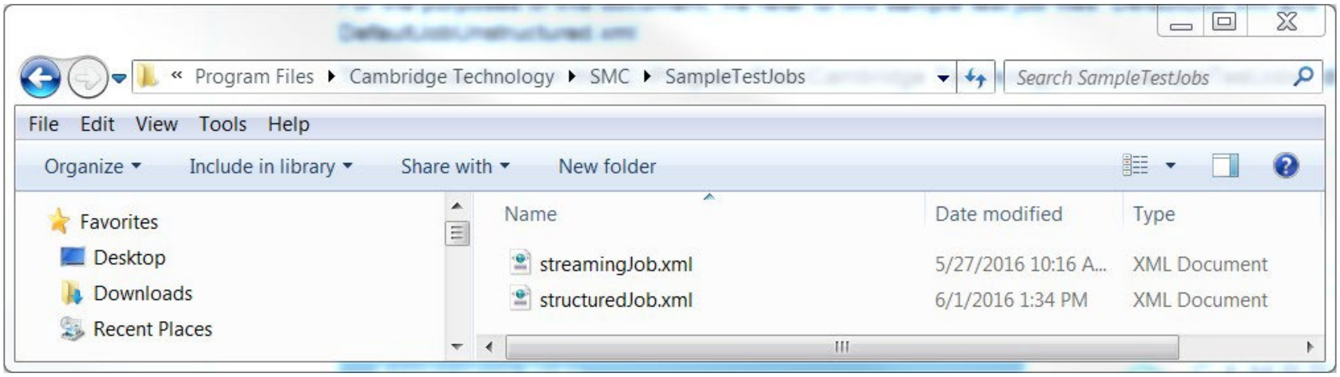


Figure 10 - **Import menu**

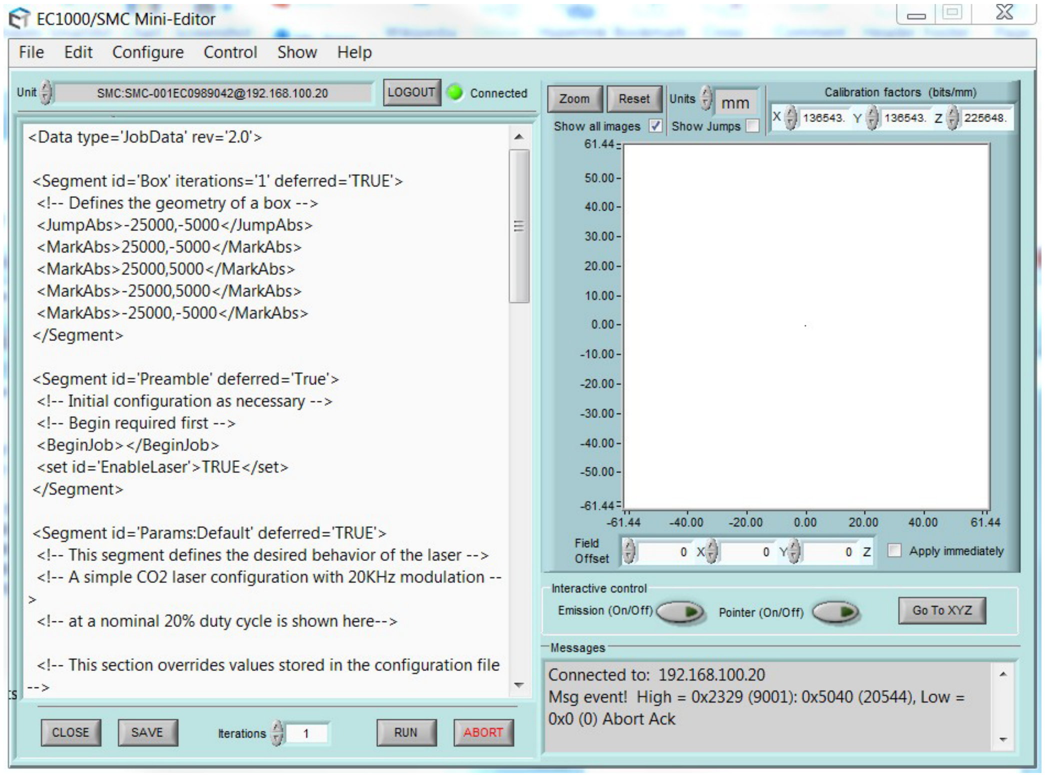
2. From Windows Explorer, browse to the job, e.g. streamingJob.xml Sample (C:\Program Files→Cambridge Technology→SMC→SampleTestJobs) and open it:



Name	Date modified	Type
streamingJob.xml	5/27/2016 10:16 A...	XML Document
structuredJob.xml	6/1/2016 1:34 PM	XML Document

Figure 11 - **Sample test jobs directory**

3. The MiniEditor should look similar to the following:



```
<Data type='JobData' rev='2.0'>

 <Segment id='Box' iterations='1' deferred='TRUE'>
  <!-- Defines the geometry of a box -->
  <JumpAbs>-25000,-5000</JumpAbs>
  <MarkAbs>25000,-5000</MarkAbs>
  <MarkAbs>25000,5000</MarkAbs>
  <MarkAbs>-25000,5000</MarkAbs>
  <MarkAbs>-25000,-5000</MarkAbs>
 </Segment>

 <Segment id='Preamble' deferred='True'>
  <!-- Initial configuration as necessary -->
  <!-- Begin required first -->
  <BeginJob></BeginJob>
  <set id='EnableLaser'>TRUE</set>
 </Segment>

 <Segment id='Params:Default' deferred='TRUE'>
  <!-- This segment defines the desired behavior of the laser -->
  <!-- A simple CO2 laser configuration with 20KHz modulation --
>
  <!-- at a nominal 20% duty cycle is shown here-->

  <!-- This section overrides values stored in the configuration file
-->
```

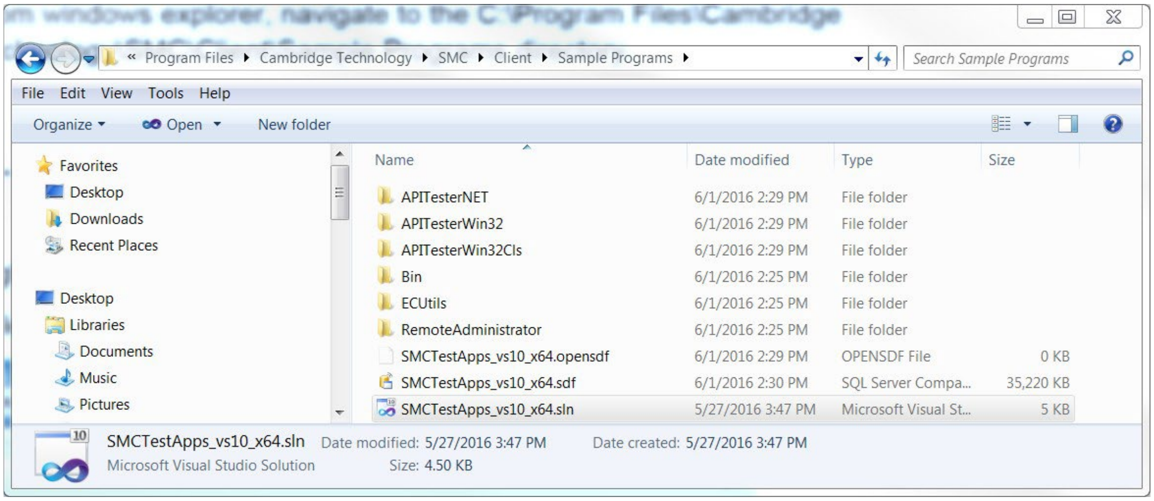
```
Connected to:  192.168.100.20
Msg event!  High = 0x2329 (9001): 0x5040 (20544), Low =
0x0 (0) Abort Ack
```

Figure 12 - **MiniEditor shown with streamingJob.xml scan job imported**

9 Example: Typical XML Job from Visual Studio 10 – SMAPI

The SMC XML API and the SMAPI together allow you to create a custom Windows application for SMC operation. The SMAPI host program is created in Microsoft Visual Studio. The SMAPI allows you to execute XML code from a SMAPI host program. Once you have installed all of the prerequisites, you can open the Visual Studio XML sample solution.

1. From Windows Explorer, navigate to the C:\Program Files\Cambridge Technology\SMC\Client\Sample Programs directory.

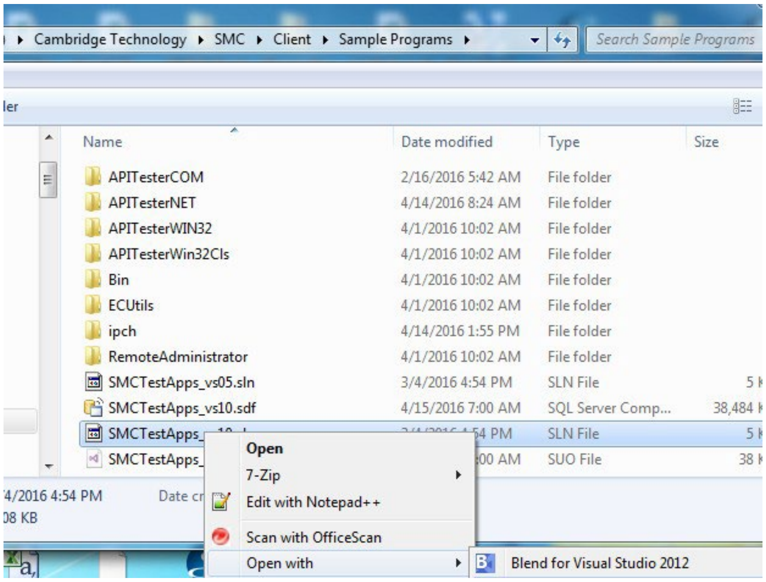

« Program Files ▸ Cambridge Technology ▸ SMC ▸ Client ▸ Sample Programs ▸
Search Sample Programs
File Edit View Tools Help
Organize ▾ Open ▾ New folder
Favorites
Desktop
Downloads
Recent Places
Desktop
Libraries
Documents
Music
Pictures

Name	Date modified	Type	Size
APITesterNET	6/1/2016 2:29 PM	File folder	
APITesterWin32	6/1/2016 2:29 PM	File folder	
APITesterWin32Cls	6/1/2016 2:29 PM	File folder	
Bin	6/1/2016 2:25 PM	File folder	
ECUtils	6/1/2016 2:25 PM	File folder	
RemoteAdministrator	6/1/2016 2:25 PM	File folder	
SMCTestApps_vs10_x64.opensdf	6/1/2016 2:29 PM	OPENSDF File	0 KB
SMCTestApps_vs10_x64.sdf	6/1/2016 2:30 PM	SQL Server Compa...	35,220 KB
SMCTestApps_vs10_x64.sln	5/27/2016 3:47 PM	Microsoft Visual St...	5 KB

SMCTestApps_vs10_x64.sln Date modified: 5/27/2016 3:47 PM Date created: 5/27/2016 3:47 PM
Microsoft Visual Studio Solution Size: 4.50 KB

Figure 13 - **Sample Programs – SMCTestApps_vs10_x64.sln directory**

2. Right mouse click on SMCTestApps_vs10_x64.sln.
3. Select open with Visual Studio 10.

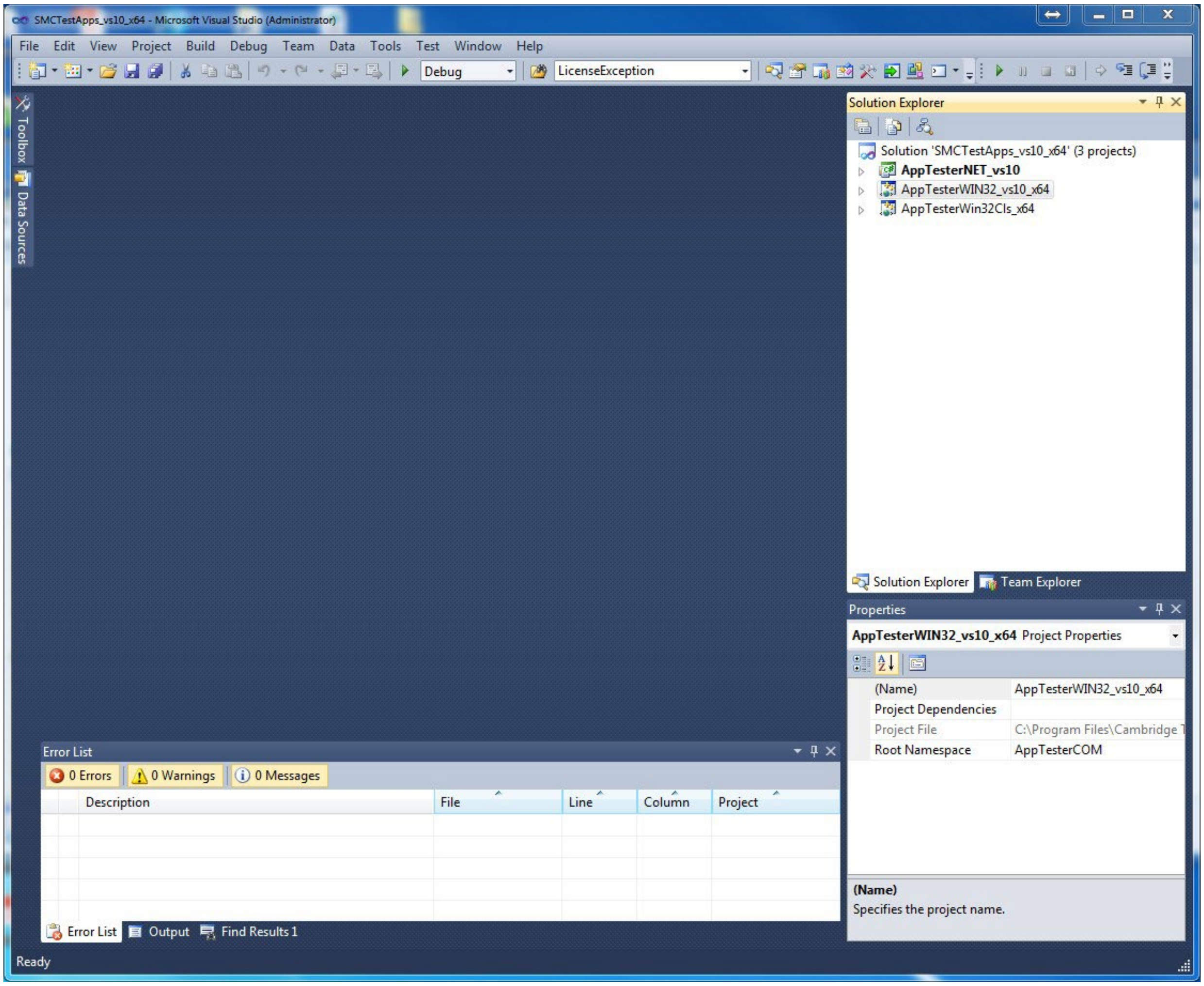

▸ Cambridge Technology ▸ SMC ▸ Client ▸ Sample Programs ▸
Search Sample Programs
ler
Open
7-Zip
Edit with Notepad++
Scan with OfficeScan
Open with
Blend for Visual Studio 2012

Name	Date modified	Type	Size
APITesterCOM	2/16/2016 5:42 AM	File folder	
APITesterNET	4/14/2016 8:24 AM	File folder	
APITesterWIN32	4/1/2016 10:02 AM	File folder	
APITesterWin32Cls	4/1/2016 10:02 AM	File folder	
Bin	4/1/2016 10:02 AM	File folder	
ECUtils	4/1/2016 10:02 AM	File folder	
ipch	4/14/2016 1:55 PM	File folder	
RemoteAdministrator	4/1/2016 10:02 AM	File folder	
SMCTestApps_vs05.sln	3/4/2016 4:54 PM	SLN File	5
SMCTestApps_vs10.sdf	4/15/2016 7:00 AM	SQL Server Comp...	38,484
SMCTestApps_	54 PM	SLN File	5
SMCTestApps_	00 AM	SUO File	38

4/2016 4:54 PM Date cr
08 KB

Figure 14 - **Open with Visual Studio**

Once Visual Studio opens, you should see:



Solution 'SMCTestApps_vs10_x64' (3 projects)
- **AppTesterNET_vs10**
- AppTesterWIN32_vs10_x64
- AppTesterWin32Cls_x64

AppTesterWIN32_vs10_x64 Project Properties

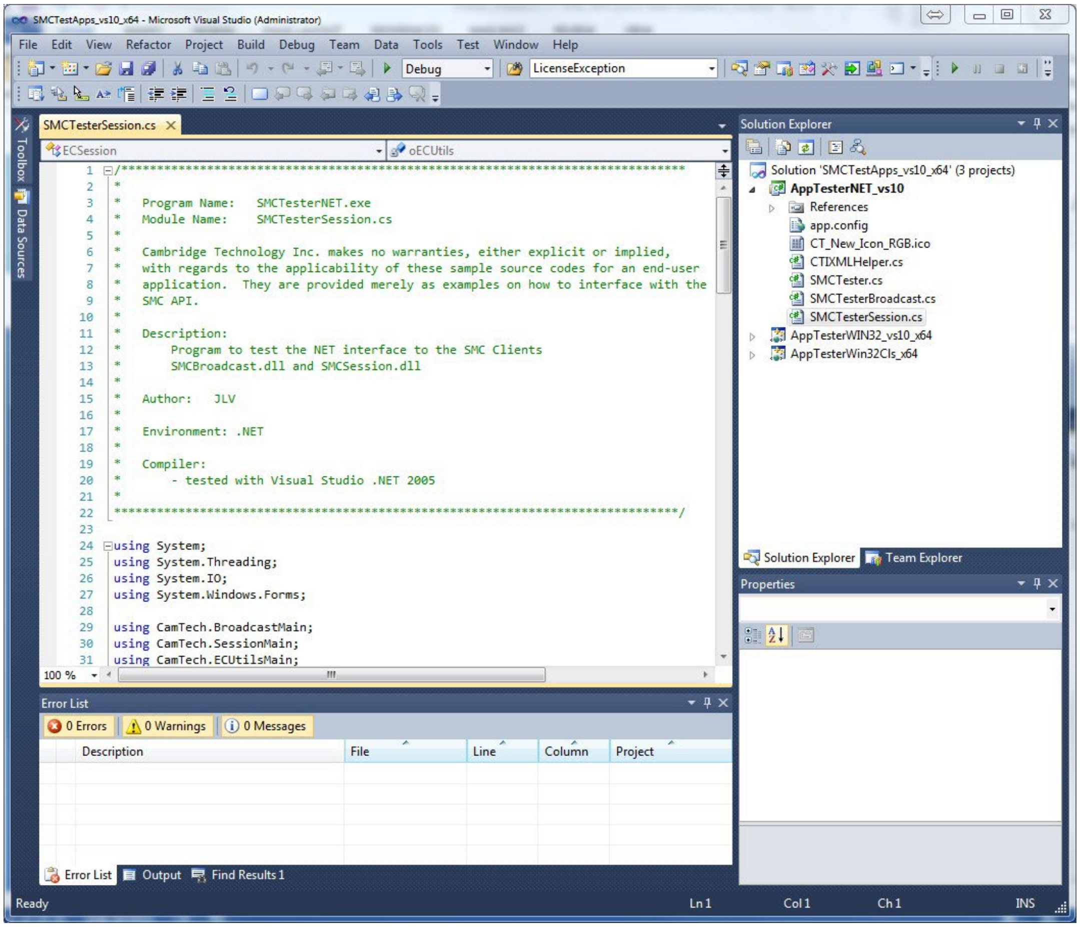
(Name)	AppTesterWIN32_vs10_x64
Project Dependencies	
Project File	C:\Program Files\Cambridge T
Root Namespace	AppTesterCOM

(Name)
Specifies the project name.

Figure 15 - **Visual Studio with SMCTestApps_vs10_x64.sln loaded**

4. In solution explorer select SMCTesterSession.cs.

You should see:

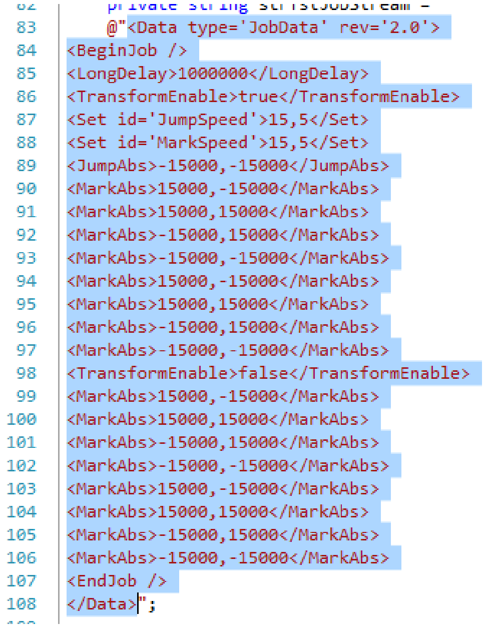


```
/*****************************************************************************
 *
 *   Program Name:   SMCTesterNET.exe
 *   Module Name:    SMCTesterSession.cs
 *
 *   Cambridge Technology Inc. makes no warranties, either explicit or implied,
 *   with regards to the applicability of these sample source codes for an end-user
 *   application.  They are provided merely as examples on how to interface with the
 *   SMC API.
 *
 *   Description:
 *       Program to test the NET interface to the SMC Clients
 *       SMCBroadcast.dll and SMCSession.dll
 *
 *   Author:   JLV
 *
 *   Environment: .NET
 *
 *   Compiler:
 *       - tested with Visual Studio .NET 2005
 *
 *****************************************************************************/

using System;
using System.Threading;
using System.IO;
using System.Windows.Forms;

using CamTech.BroadcastMain;
using CamTech.SessionMain;
using CamTech.ECUtilsMain;
```

Figure 16 - **Visual Studio with SMCTesterSession.cs**

5. Scroll down to line 83 and select from after the " character to before the " character on line 108.

You should see something like:



```
 82         private string strFirstJobStream =
 83         @"<Data type='JobData' rev='2.0'>
 84     <BeginJob />
 85     <LongDelay>1000000</LongDelay>
 86     <TransformEnable>true</TransformEnable>
 87     <Set id='JumpSpeed'>15,5</Set>
 88     <Set id='MarkSpeed'>15,5</Set>
 89     <JumpAbs>-15000,-15000</JumpAbs>
 90     <MarkAbs>15000,-15000</MarkAbs>
 91     <MarkAbs>15000,15000</MarkAbs>
 92     <MarkAbs>-15000,15000</MarkAbs>
 93     <MarkAbs>-15000,-15000</MarkAbs>
 94     <MarkAbs>15000,-15000</MarkAbs>
 95     <MarkAbs>15000,15000</MarkAbs>
 96     <MarkAbs>-15000,15000</MarkAbs>
 97     <MarkAbs>-15000,-15000</MarkAbs>
 98     <TransformEnable>false</TransformEnable>
 99     <MarkAbs>15000,-15000</MarkAbs>
100     <MarkAbs>15000,15000</MarkAbs>
101     <MarkAbs>-15000,15000</MarkAbs>
102     <MarkAbs>-15000,-15000</MarkAbs>
103     <MarkAbs>15000,-15000</MarkAbs>
104     <MarkAbs>15000,15000</MarkAbs>
105     <MarkAbs>-15000,15000</MarkAbs>
106     <MarkAbs>-15000,-15000</MarkAbs>
107     <EndJob />
108     </Data>";
```

Figure 17 - **Line 83–108 from SMCTesterSession.cs**

You can replace the selected XML inside the string with your own SML code, allowing you to host your XML code inside a C# SMAPI host Windows application.

NOTE: SMCTesterSession is part of a larger C# sample solution installed with SMAPI.

You can use this sample as a starting point for your own Windows hosted applications.

If you want to use C++, you can use the AppTesterWin32Cls_vs10 project.

Once you have the XML code in the sample, you can run the application.

6. Press start in Visual Studio.

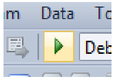
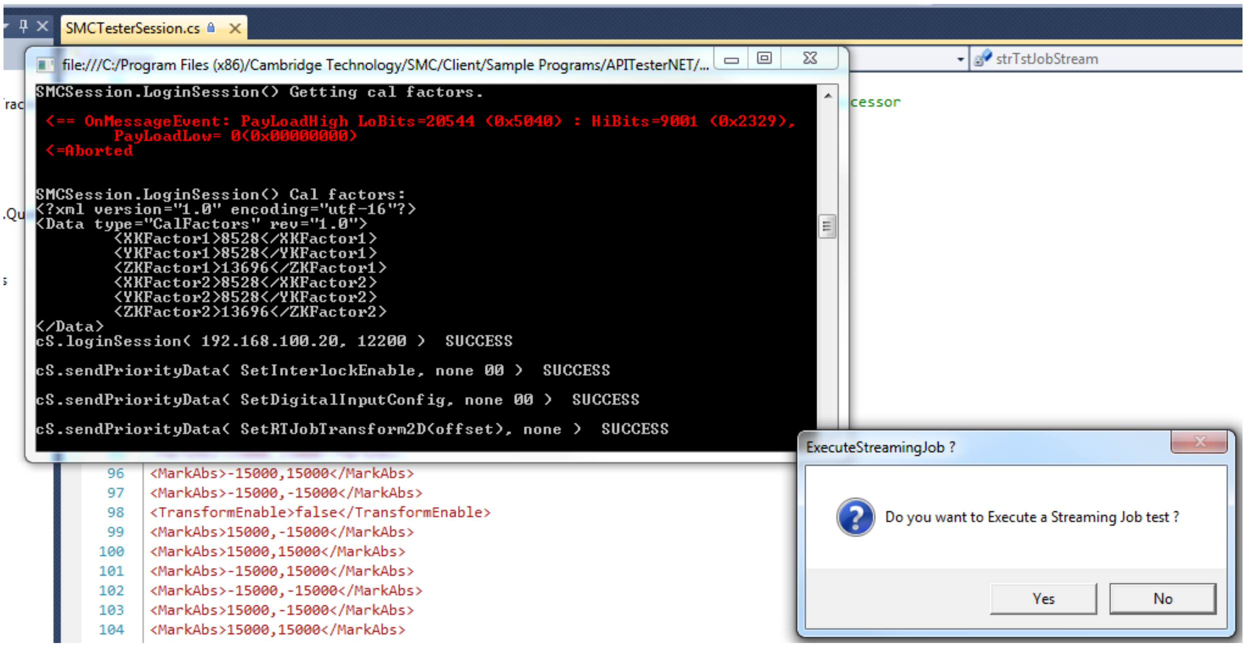


Figure 18 - **Start button in Visual Studio**


SMCTesterSession.cs
file:///C:/Program Files (x86)/Cambridge Technology/SMC/Client/Sample Programs/APITesterNET/...
strTstJobStream
ExecuteStreamingJob ?
Yes
No

```
SMCSession.LoginSession() Getting cal factors.
 <== OnMessageEvent: PayLoadHigh LoBits=20544 (0x5040) : HiBits=9001 (0x2329),
        PayLoadLow= 0(0x00000000)
 <=Aborted


SMCSession.LoginSession() Cal factors:
<?xml version="1.0" encoding="utf-16"?>
<Data type="CalFactors" rev="1.0">
        <XKFactor1>8528</XKFactor1>
        <YKFactor1>8528</YKFactor1>
        <ZKFactor1>13696</ZKFactor1>
        <XKFactor2>8528</XKFactor2>
        <YKFactor2>8528</YKFactor2>
        <ZKFactor2>13696</ZKFactor2>
</Data>
cS.loginSession( 192.168.100.20, 12200 )  SUCCESS

cS.sendPriorityData( SetInterlockEnable, none 00 )  SUCCESS

cS.sendPriorityData( SetDigitalInputConfig, none 00 )  SUCCESS

cS.sendPriorityData( SetRTJobTransform2D(offset), none )  SUCCESS
```

```
<MarkAbs>-15000,15000</MarkAbs>
<MarkAbs>-15000,-15000</MarkAbs>
<TransformEnable>false</TransformEnable>
<MarkAbs>15000,-15000</MarkAbs>
<MarkAbs>15000,15000</MarkAbs>
<MarkAbs>-15000,15000</MarkAbs>
<MarkAbs>-15000,-15000</MarkAbs>
<MarkAbs>15000,-15000</MarkAbs>
<MarkAbs>15000,15000</MarkAbs>
```

Do you want to Execute a Streaming Job test ?

Figure 19 - **Streaming job test**

You can click "No" to all the Dialog boxes that pop up, but click "Yes" for "Do you want to execute a streaming job test?"

You can click "No" to all the remaining dialogs to end the application.

10 The Host Application and the CTIXMLHelper Class

This document refers to the Cambridge Technology sample: SMCTestApps_vs10_x64.sln. It demonstrates use of the CTIXMLHelper.cs class, which simplifies the creation of underlying XML and packages some of the coding complexities into an easier interface to help prevent common errors.

In this example:

1. The Visual Studio sample application connects with a session to an SMC controller that is broadcasting on the network.
2. Once connected, the XML job can be streamed on the controller for execution.
3. When the XML job is done executing, the connection can disconnect and the application can stop listening to broadcasting controllers.
4. The application can then exit.

Following is a description of key points in the SMCTester.cs file (found in solution explorer) that apply to the above typical scenario. You can use the code as a template for your application development.

NOTE: All line numbers below refer to the SMCTester.cs file.

5. **SMCTester.cs line 63** – Open a Broadcast Object to listen to broadcasting SMC controllers.
6. **Lines 69–113** – Generate a list of all available SMC controllers. In SMCTester.cs, the code gets available broadcast data for each controller.

7. **Lines 116–133** – If there is more than one SMC controller, select the controller you want to use. If there is just one controller, use device 0 (the first device found).
8. **Lines 125–140** – Next create a session and connect the selected SMC to the session.
9. Once you have a session and a connection to the SMC controller, use the SendStreamData method to stream commands to the SMC.

SendStreamData takes the XML command job data in the form of a data string and sends it to the SMC controller for processing. You can use repeated calls to SendStreamData to keep the buffer full and allow the SMC controller to process commands until the job is complete.

10.1 Sample Code Segment

The following code segment (SMCTesterSession.cs Lines 603–621) generates a proper XML data job using the CTIXMLHelper class. It generates a XML job similar to the pure XML job above shown in the DefaultJobUnstructured.xml MiniEditor example.

```
CTIXmlHelper jhp = new CTIXmlHelper(cS);
jhp.EC_StartPacket();
jhp.EC_BeginJob();
jhp.EC_SetActuatorUnits(ActuatorUnits.bits20);
jhp.EC_SetLaserTimingResolution(0.1F);
jhp.EC_SetLaserTiming(periodInUsec, pulse1InUsec, pulse2InUsec);
jhp.EC_SetScannerDelays(jumpdelayInUsec, markdelayInUsec,
   polydelayInUsec);
jhp.EC_SetLaserDelays(OnDelayInUsec,OffDelayInUsec,ModDelayInUsec,
   PipelineDelayInUsec);
jhp.EC_SetJumpSpeed(fJumpSpeedInBitsPerMsec);
jhp.EC_SetMarkSpeed(fMarkSpeedInBitsPerMsec);
jhp.EC_JumpAbsEx(-10000.0f, -10000.0f, 0.0f);
jhp.EC_MarkAbsEx(10000.0f, -10000.0f, 0.0f);
jhp.EC_MarkAbsEx(10000.0f, 10000.0f, 0.0f);
jhp.EC_MarkAbsEx(-10000.0f, 10000.0f, 0.0f);
jhp.EC_MarkAbsEx(-10000.0f, -10000.0f, 0.0f);
jhp.EC_EndJob();
jhp.EC_EndPacket();
hr = jhp.EC_SendPacket();
```

The actions for each line are described below:

1. Creates a CTIXMLHelper object (jhp)
2. Generates a <Data type='JobData' rev='2.0'> XML tag and appends it to the jhp job
3. Generates a <BeginJob /> XML tag and appends it to the jhp job
4. Generates a <ActuatorUnits>**bits-20**</ActuatorUnits> tag and appends it to the jhp job
5. Generates a <set id='LaserTiming'>5</set> tag and appends it to the jhp job
6. Generates a <set id='LaserPulse'> 1,50,100</set>\n<set id='LaserPulse'>2,50,100</set> tag and appends it to the jhp job
7. Generates <set id='JumpDelay'> 200</set>\n<set id='MarkDelay'> 200</set>\n<set id='PolyDelay'> 75</set> tags and appends it to the jhp job
8. Generates <set id='LaserOnDelay'>75</set><set id='LaserOffDelay'>150</set><set id='LaserModDelay'>2</set><set id='LaserPipelineDelay'>0</set>> tags and appends it to the jhp job
9. Generates <set id='JumpSpeed'> 500,500 </set> tag and appends it to the jhp job
10. Generates <set id='MarkSpeed'> 10,10</set> tag and appends it to the jhp job
11. Generates <JumpAbsEx>**-10000; -10000; 0**</JumpAbsEx> tag and appends it to the jhp job
12. Generates <MarkAbsEx>**-10000; -10000; 0**</MarkAbsEx> tag and appends it to the jhp job
13. Generates <MarkAbsEx>**-10000; 10000; 0**</MarkAbsEx> tag and appends it to the jhp job
14. Generates <MarkAbsEx>**-10000; 10000; 0**</MarkAbsEx> tag and appends it to the jhp job
15. Generates <MarkAbsEx>**-10000; -10000; 0**</MarkAbsEx> tag and appends it to the jhp job
16. Generates <EndJob></EndJob> tag and appends it to the jhp job
17. Generates <ApplicationEvent>1,1</ApplicationEvent></Data> tags and appends it to the jhp job
18. Calls session.sendStreamData(m_jobList.ToString(), CTI_TRANSACTION_TIMEOUT) SendPacket calls sendStreamData

When the job has completed:

- Log out of the session by calling *iSession.Logout()* (See SMCTester.cs line 145), which calls *cS.logoutSession(SMC_TIMEOUT).*
- Finally, call iBroadcast.Detach() to stop the application from listening to the broadcasts.