Working with Print Job Status Events

The BarTender Print SDK provides several events used to monitor a print job's status. When any label format is opened, various Windows spooler events may be monitored. Events may be handled for a specific `LabelFormatDocument` instance or across the entire `Engine` object. Classes in Print Server namespace also implement these same events: `XMLScriptTask` and `PrintLabelFormatTask`. Working with print job status events in Print Server SDK is fundamentally the same process as presented here.

Job status events provided by the BarTender Print SDK include:

- `JobCancelled`
- `JobErrorOccurred`
- `JobMonitorErrorOccurred`
- `JobPaused`
- `JobQueued`
- `JobRestarted`
- `JobResumed`
- `JobSent`

How Job Status Events Work

The job status events depend on the Maestro Print Service. To use the job status events, the Printer Maestro service must be installed. If printing to a networked printer, the Printer Maestro service must be installed on both the computer to which the networked printer is attached and the local machine running the BarTender .Net SDK application.

Generally, job status events are called asynchronous to the thread originating a print action. This means an application that uses `Forms` cannot assume job status events will be called on the UI thread; facilities such as **`Invoke`** and **`BeginInvoke`** must be used to modify UI objects from event handling methods.

Subscribing to Job Status Events

`LabelFormatDocument` and the `Engine` class both implement the same set of job monitoring events, allowing print job monitoring for either a particular `LabelFormatDocument` instance's print jobs or for all print jobs generated by an `Engine` instance.

The following is an example demonstrating how to subscribe to a `LabelFormatDocument`'s print jobs.

In C#:

```
// Open a label format
LabelFormatDocument btFormat = btEngine.Documents.Open
(@"c:\MyLabelFormat.btw");
// Subscribe to the format event
btFormat.JobQueued += new EventHandler<PrintJobEventArgs>
(MyLabelFormatOnJobQueued);
```

In VB:

```
' Open a label format
```

```
Dim btFormat As LabelFormatDocument = btEngine.Documents.Open
("c:\MyLabelFormat.btw")
' Subscribe to the format event
AddHandler btFormat.JobQueued, AddressOf MyLabelFormatOnJobQueued
```

In the above code, a new label format is opened. The `JobQueued` event is subscribed to using an event handler which accepts instances of the `PrintJobEventArgs` class.

Handling Job Status Events

When handling a job status event, the sender is always the object on which the event was raised. The argument varies by the specific event and is one of the following types: `PrintJobEventArgs`, `JobSentEventArgs`, or `MonitorErrorEventArgs`. Each of the argument classes is descended from `EventArgs`, meaning it is also possible to service event using an instance of `System.EventHandler` in addition to the generic `System.EventHandler<T>`.

The following code example shows how to handle the `JobQueued` event when serviced from an instance of the `Engine` class:

In C#:

```
// Called by a worker thread when a job is queued
public void btEngine_JobQueued(object sender, PrintJobEventArgs e)
{
  Engine btEngine = sender as Engine;

  if (btEngine != null)

  {

    Console.WriteLine(string.Format("Client {0} has printed or is printing {1}
    jobs", e.ClientName, btEngine.PrintJobCounter));

  }

}
```

In VB:

```
' Called by a worker thread when a job is queued
Public Sub btEngine_JobQueued(ByVal sender As Object, ByVal e As
PrintJobEventArgs)
  Dim btEngine As Engine = TryCast(sender, Engine)

  If btEngine IsNot Nothing Then

    Console.WriteLine(String.Format("Client {0} has printed or is printing {1}
    jobs", e.ClientName, btEngine.PrintJobCounter))

  End If

End Sub
```

The above code casts the sender into an `Engine`. It then prints to console the name of the sending computer and how many print jobs the firing engine instance has performed.

Subscribing to Job Status events when using BTXML Script

When using the **XMLScript** method of an Engine object, the job status events of the Engine instance will be fired. Because there are no instances of LabelFormatDocument involved with the submission of the BTXML script, there will be no corresponding instance of LabelFormatDocument whose events are fired.

Use of the XMLScript command requires the Enterprise Automation edition of BarTender.

Engine Events

Engine events provide the ability to monitor the status of any print job controlled by the Engine instance. These events may be monitored for any label format opened by the engine.

Aside from providing numerous events that are also provided in the LabelFormatDocument class, the Engine class provides another event: CommandLineCompleted. This event is only available in the BarTender Enterprise Automation edition, and provides a method of monitoring automation through the use of command line parameters.

Format Events

Format events provide the ability to monitor the status of a print job specific to a LabelFormatDocument object. These events may be subscribed to, but only for a specific label format. Excluding the Engine class' CommandLineCompleted event, the events for LabelFormatDocument are identical to those provided by Engine.

Using an Event

The following is an example of event handling using events in LabelFormatDocument:

In C#:

```
public void ProgramModule()
{
  // Initialize and start a BarTender Engine

  using (Engine btEngine = new Engine(true))

  {

    // Open a label format

    LabelFormatDocument btFormat = btEngine.Documents.Open
    (@"c:\MyLabelFormat.btw");

    // Subscribe to the format event

    btFormat.JobQueued += new EventHandler<PrintJobEventArgs>
    (MyLabelFormatOnJobQueued);

    // Print the label

    btFormat.Print();

    // Stop the BarTender Engine

    btEngine.Stop(SaveOptions.DoNotSaveChanges);

  }

}
public void MyLabelFormatOnJobQueued(object sender, PrintJobEventArgs
printJobEventInfo)
{
  // Cast the sender into a Format object
```

```
    LabelFormatDocument btFormat = sender as LabelFormatDocument;

    // Check to see if the object is not null

    // If it is, the sender was NOT a Format

    if (btFormat != null)

    {

      // Do something with the label format

    }

  }
```

In VB:

```
  Public Sub ProgramModule()
    ' Initialize and start a BarTender Engine

    Using btEngine As New Engine(True)

      ' Open a label format

      Dim btFormat As LabelFormatDocument = btEngine.Documents.Open
      ("c:\MyLabelFormat.btw")

      ' Subscribe to the format event

      AddHandler btFormat.JobQueued, AddressOf MyLabelFormatOnJobQueued

      ' Print the label

      btFormat.Print()

      ' Stop the BarTender Engine

      btEngine.Stop(SaveOptions.DoNotSaveChanges)

    End Using

  End Sub
  Public Sub MyLabelFormatOnJobQueued(ByVal sender As Object, ByVal
  printJobEventInfo As PrintJobEventArgs)
    ' Cast the sender into a Format object

    Dim btFormat As LabelFormatDocument = TryCast(sender, LabelFormatDocument)

    ' Check to see if the object is not null

    ' If it is, the sender was NOT a Format

    If btFormat IsNot Nothing Then

      ' Do something with the label format

    End If

  End Sub
```

The above code demonstrates how to open, print, and close a label using events. After a format is opened and initialized, a new delegate subscribes to the `JobQueued` event. When the job is sent from BarTender to the spooler queue, the `JobQueued` event is raised. Inside the event handler, the object sender is cast into a `LabelFormatDocument`. The sender in this case is the label format on which the **`Print`** method was called.