Commander Examples

Step-by-Step Instructions on How to Use Commander in a Variety of Common Scenarios

SEAGULL SCIENTIFIC

Contents

The purpose of this paper is to present detailed examples of the most common ways that Seagull Scientific's Commander Middleware Integration Utility is put to use. Step-by-step instructions for implementing each of these examples are included.

What is Commander?

Commander is a software utility, available with the two Automation editions of BarTender, that enables you to perform automatic label-printing using BarTender in situations where using command line or ActiveX automation is either not possible or desirable. Commander can be run as an application or as a Windows service.

How Commander Works

When an application needs labels, it simply creates a triggering event (called a "trigger"), such as placing a file in a location of your choosing on the network, sending an e-mail to an account of your choosing, or sending data through a TCP/IP socket. Commander detects the arrival of this trigger and then "wakes up" BarTender so it can pour your data into the label design and automatically print your labels.

A trigger file or message may be empty; but it can also contain data that will be read by BarTender, by another application that Commander launches, or by Commander itself. Because trigger-creating applications can include different content in different triggers, the tasks Commander will execute, and the data it uses, can be defined dynamically by the application.

For example, an in-house order fulfillment application enters data about an order into a database and saves an empty file named NewOrder.dat in a directory being watched by Commander. Commander finds the file and launches BarTender with a label format that has been configured to query the database about orders entered after a specified time. BarTender reads the data and prints the label. Commander then deletes the NewOrder.dat file and resumes monitoring the directory.

For a more detailed overview of Commander see Seagull Scientific's white paper *Commander: Middleware for Automatically Printing Labels in Response to User-Defined Events (No programming required)*.

Goal

The purpose of this example is to show, in detail, how Commander is configured to detect an empty trigger file or e-mail message and then to react by instructing BarTender to launch and print a specified label format. The techniques involved are also applicable to the later examples.

Case Scenario

At a parts-manufacturing plant there is a need to have a label with a serialized number printed at the end of an assembly line. On this label there is no other dynamic or changing information, other than the incremented bar code. What generates the trigger file is a "Laser Presence Sensor" toward the end of the assembly line that detects parts and generates an empty text file into a specific directory that Commander is configured to monitor.

To Implement this Scenario

Create the Label Format

> **Note**: The files created in this example can be found in 1stExample.zip inside CommanderExamples.zip located at:
>
> *ftp://ftp.seagullscientific.com/BarTender/Examples/Commander*
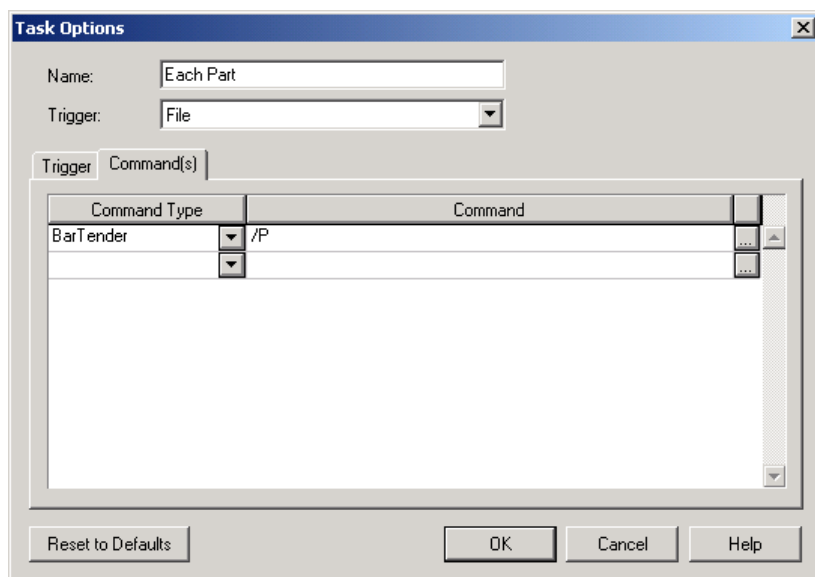>
> Use them to verify that the files you create are correct.

1. In BarTender, select **New** on the **File** menu.

2. Select **Print** on the **File** menu and select the printer you want to use from the **Name** drop down list and then click **Close**.

3. Select **Page Setup** on the **File** menu and choose a stock or configure your page as needed.

4. Create a bar code object on the label and double-click it to open the **Modify Selected Text Object** dialog.

5. Click the **Data Source** tab.

6. Click the **More Options** button and click the **Serialization** tab.

7. Check the **Serialization** check box and click the **Close** button and then the **OK** button.

8. Select **Save** in the **File** menu to save the label format under the name Serialized.btw.

9. Close BarTender

Create the Commander Task

1. Choose the network directory where the trigger files will be saved. This is the directory that Commander will monitor. In this example, it will be called **\\AssemblyLine\NextPart**.
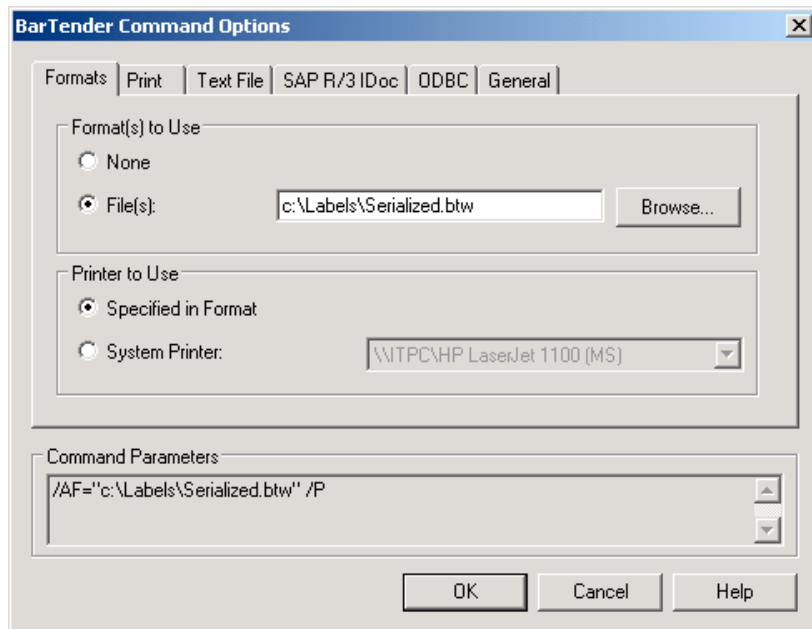
2. Create, *in some other directory*, a blank text file called "part.txt".

3. In Commander, select **New** on the **File** menu.

4. Select **Add** on the **Task** menu to open the **Task Options** dialog.

5. Enter "Each Part" (without the quotation marks) as the task **Name** and select **File** in the **Trigger** drop down list.

6. Click the **Trigger** tab, and in the **Folders to Scan** box, enter the path of the directory that Commander will monitor. This is the same directory that you used in step 1 above.

   Task Options

   Name: New Item
   Trigger: File

   Trigger | Command(s)

   Detection Method: Immediate
   Polling Interval: 1000 Milliseconds
   Folder to Scan: \\Inventory\IDocs\   Browse...
   Files to Scan for: IDD-LIEF03_00108227.WTADDI0_46C
   Detection Response: Rename File
   New Extension: old
   Locked-File Timeout: 30 Seconds

   Reset to Defaults   OK   Cancel   Help

7. Enter "part.txt" or just "*.txt" in the **Files to Scan for** box. (Do not include the quotation marks.)

8. Set the **Detection Response** to **Delete File**, because BarTender will not need to open and read the file.

   Task Options

   Name: Each Part
   Trigger: File

   Trigger | Command(s)

Command Type	Command
BarTender	/P

   Reset to Defaults   OK   Cancel   Help

9. Click the **Commands** tab and in the first row of the **Command Type** column, select **BarTender** from the drop down list.

10. Click the button labeled "**…**" at the end of the row to open the **BarTender Command Options** dialog. Note that there is initially just a `/P` parameter in the **Command Parameters** box. This tells BarTender to print.

11. Click the **Formats** tab and select the **File(s)** radio button. Then enter the path and file name for Serialized.btw. Note that an `/AF=` parameter has been added to the **Command Parameters** box identifying for BarTender which label format it should print.

12. Click **OK** on the **BarTender Command Options** dialog, and then click **OK** on **Task Options** dialog.

13. Select **Save** in the **File** menu and save the task list under the name NewPart.tl.

Start Detection

1. Select **Start Detection** from the **Detection** menu.

2. Test your configuration by copying the sample file, part.txt, into the directory that Commander is monitoring.

Commander will:
- detect the file part.txt,
- delete it, and
- launch BarTender.

Bartender will:
- open Serialized.btw, and
- print the label.

Now you can have an inventory tracking application, or some other application, automatically create files named part.txt in the directory that Commander is monitoring and Bartender will print out this label each time such a file is created.

Goal

The goal of this example is to show, in detail, how Commander can use the trigger file, or message, itself as a data file. The file can contain one or more rows of delimited (or fixed-width) fields of data that Commander reads and passes to BarTender. The latter uses the data to populate the fields on the label at print-time.

Case Scenario

At a mail order company there is a need to print a label containing data from a sales record each time an order is packaged. The sales application generates a comma-delimited text file and saves it in a directory that Commander has been set to monitor. Commander detects the file and instructs BarTender to get data from the file and print the label.

To Implement this Scenario

Set Up a Sample Data File

Note: The files created in this example can be found in 2ndExample.zip inside CommanderExamples.zip located at:

ftp://ftp.seagullscientific.com/BarTender/Examples/Commander

Use them to verify that the files you create are correct.

1. Choose the network directory where the trigger files will be saved. This is the directory that Commander will monitor. In this example, it will be called
   `\\ShippingServer\NewOrders`

2. Create, or obtain, a sample file that is structured and delimited exactly as the real data files will be. In this example, we will create a file in Notepad that will be named order.txt. It will contain a single line of comma-delimited data (as shown below). There are no hard carriage returns in the file.

   ```
   Around the Town,Jerome Davis,Buyer,1220 Governor
   Sq.,Hadden, Alberta,WA1 1DP,Canada,(171) 555-7788,(171)
   555-6750
   ```

3. Save the sample file in the directory that you chose in step 1.
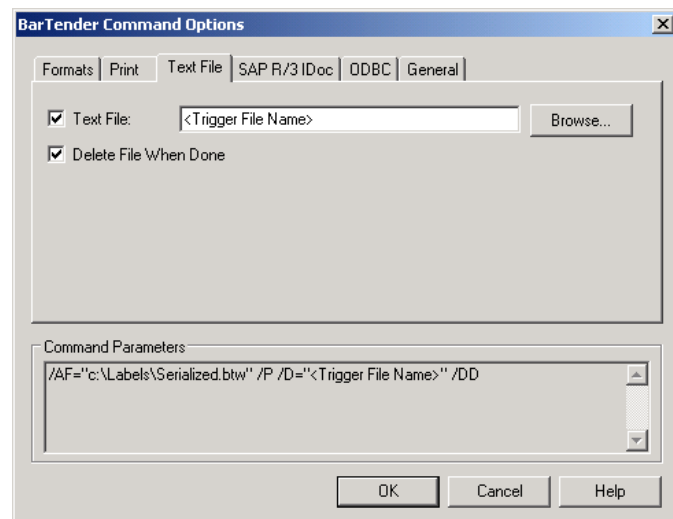
Create the Label Format

1. In BarTender, select **New** on the **File** menu.

2. Select **Print** on the **File** menu and select the printer you want to use from the **Name** drop down list and then click **Close**.

3. Select **Page Setup** on the **File** menu and choose a stock or configure your page as needed.

4. Click the **Database Connection Setup** button.

5. In the **Add Database Connection Wizard**, click **Next**.

6. Select **Text File** and click **Next**.

7. Enter the complete path and file name of the sample file you created, and then click **Next**.

8. Choose **Comma** as the **Delimitation Type**, and then click **Next**.

9. Select **No** when asked about the text file header, and then click **Finish** and then click **OK** to close the **Database Connection Setup** dialog.

10. Create a text object on the label and double-click it to open the **Modify Selected Text Object** dialog.

11. Click the **Data Source** tab and set the **Source** drop down list to **Database Field**.

12. On the **Use Field** drop down list, select "Field 2" in the **Use Field** drop down list because, in the sample, the second field contains the buyer's name.

13. Enter "buyer name" as the sample data, and click **OK** to close the dialog.

14. If other text objects are needed, simply return to step 11, in each case assigning the object to the appropriate field in the database, otherwise continue onto step 16.

15. Select **Save** in the **File** menu to save the label format under the name NewOrderAddress.btw.

16. Move the sample database file, orders.txt, from the directory that Commander will be scanning.

17. Close BarTender.

Create the Commander Task

1. In Commander, select **New** on the **File** menu.

2. Select **Add** on the **Task** menu to open the **Task Options** dialog.

3. Enter "Each Order" (without the quotation marks) as the task **Name** and select **File** in the **Trigger** drop down list.

4. Click the **Trigger** tab, and in the **Folders to Scan** box, enter the path of the directory that Commander will monitor. This is the same directory that you used in step 1 of *Set Up a Sample Data File* above.

5. Enter "order.txt" or just "*.txt" in the **Files to Scan for** box. (without the quotation marks.)

6. Set the **Detection Response** to **Rename File**, rather than **Delete File** because BarTender cannot use the file as its database if Commander has deleted it.

7. Set the **New Extension** to "old" (without the quotation marks). This extension cannot be the same one specified in **Files to Scan for**.

8. Click the **Commands** tab and in the first row of the **Command Type** column, select **BarTender** from the drop down list.

9. Click the button labelled "**…**" at the end of the row to open the **BarTender Command Options** dialog. Note that there is initially just a `/P` parameter in the **Command Parameters** box. This tells BarTender to print.

10. Click the **Formats** tab and select the **File(s)** radio button. Then enter the path and file name for NewOrderAddress.btw. Note that an `/AF=` parameter has been added to the **Command Parameters** box identifying for BarTender which label format it should print.

11. Click the **Text File** tab and enable the **Text File** check box, but leave the value at its default, the Commander variable %Trigger File Name%. A `/D=%Trigger File Name%` parameter has been added to the **Command Parameters** box telling BarTender that it should use the trigger file as its database.

12. Enable the **Delete File When Done** checkbox to tell BarTender that it should delete the file after reading it. A `/DD` parameter has been added to the **Command Parameters** box telling BarTender that it should delete the trigger file after reading it.

13. Click **OK** on the **BarTender Command Options** dialog, and then click **OK** on the **Task Options** dialog.

14. Select **Save** in the **File** menu and save the task list under the name NewOrderAddress.tl.

Start Detection

1. Select **Start Detection** from the **Detection** menu.

2. Test your set up by copying the sample file, orders.txt, back into the directory that Commander is monitoring.

Commander will:

- detect the file orders.txt,

- rename it to orders.old, and

- launch BarTender.

Bartender will:

- open NewOrderAddress.btw,

- read data from orders.old,

- print the label, and

- delete orders.old.

Now you can have a sales-entry application, or some other application, automatically create files named orders.txt (containing the data from the sale) in the directory that Commander is monitoring and Bartender will print out this label for each such file.

Goal

The goal of this example is to explain how to include a special Commander script in a trigger file or message and configure Commander to use it.

Some companies have different applications and databases with label-printing needs, scattered throughout their network. In such cases, the label format that should be printed can be varied depending on the contents of the trigger file or message.

With little programming these applications can be modified or configured to write data and command lines to a text file. That is why there is an option to have a Commander script embedded in the trigger file along with the data. A Commander script is a set of written instructions that Commander can read and execute.

This feature enables the application to give Commander instructions that vary from trigger-to-trigger, instead of having it perform exactly the same actions in response to every trigger of a given type.

Case Scenario

A company needs to print sticky-backed invoices being entered by many different users across their network. Each user has access to a shared drive. However, there are differently formatted invoices depending on the user and the product being invoiced. And the various departments each use their own printer.

In this situation, an invoicing application can create, with each invoice, a text file containing data and a Commander script that instructs BarTender to:

- open up a particular invoice *.btw file,

- use the trigger file as the data file, starting on line 3,

- print to a specific printer, and

- delete the trigger file when it's done.

To Implement this Scenario

Set Up Sample Data Files

Note: The files created in this example can be found in 3rdExample.zip inside CommanderExamples.zip located at:

ftp://ftp.seagullscientific.com/BarTender/Examples/Commander

Use them to verify that the files you create are correct.

1. Choose the network directory where the trigger files will be saved. This is the directory that Commander will monitor. In this example, it will be called **\\InvoiceServer\CommanderWatch**

2. For each type of invoice, create, or obtain, a sample file in which the data is structured and delimited exactly as it will be in the real data files. In this example, we will assume that there are two departments printing invoices. For the first, create a file in Notepad named invoice1.txt that contains a single line of comma-delimited data (as shown below). The only hard carriage return is at the end of the record.

   ```
   Around the Town,Jerome Davis,Buyer,1220 Governor Sq.,Hadden,
   Alberta,WA1 1DP,Canada,(171) 555-7788,(171) 555-6750
   ```

3. For the second, create a file in Notepad named invoice2.txt that contains a single line of quote-and-comma-delimited data (as shown below). The only hard carriage return is at the end of the record.

   ```
   "Far Fetched Imports","Jerry Bom","manager","3320 Happy
   St.","Incident","Washington","98887","USA","(444) 555-
   7788","(444) 555-6750"
   ```

4. Save the sample files in the directory that you chose in step 1.

Create the Label Formats

1. Following the same steps as the *Create the Label Format* section of the previous example, create a label format for the first department, using invoice1.txt as the database. Use HardwareInvoice.btw as the label format's file name.

2. Create a second label format for the second department, using invoice2.txt as the database. (Be sure to indicate that the delimitation type is quote-and-comma.) Use SoftwareInvoice.btw as the label format's file name.

3. Move both sample database files from the directory that Commander will be scanning.

4. Close BarTender.

Create the Commander Task

1. In Commander, select **New** on the **File** menu.

2. Select **Add** on the **Task** menu to open the **Task Options** dialog.

3. Enter "Invoices" (without the quotation marks) as the task **Name** and select **File** in the **Trigger** drop down list.

4. Click the **Trigger** tab, and in the **Folders to Scan** box, enter the path of the directory that Commander will monitor. This is the same directory that you used in step 1 of *Set Up Sample Data Files* above.

5. Enter "*.txt" in the **Files to Scan for** box. (Do not include the quotation marks.)

6. Set the **Detection Response** to **Rename File**, rather than **Delete File** because BarTender cannot use a file as a database if Commander has deleted it.

7. Set the **New Extension** to "old" (without the quotation marks). This extension cannot be the same one specified in **Files to Scan for**.

8. Click the **Commands** tab and in the first row of the **Command Type** column, select **Commander Script** from the drop down list.

9. Click **OK** on the **Task Options** dialog.

10. Select **Save** in the **File** menu and save the format under the name VariantInvoices.tl.

Creating Sample Trigger Files

1. Open the first sample database file, invoice1.txt, and add the following two lines to the very beginning of the file. (The only hard carriage returns are immediately before and immediately after the "`%END%`".)

```
%BTW% /AF=c:\command\command.btw /D=%Trigger File Name%
/PRN="EasyCoder F4 (203 dpi)" /R=3 /P /DD

%END%
```

So the entire file now contains the following:

```
%BTW% /AF=c:\command\HardwareInvoice.btw /D=%Trigger File
Name% /PRN="EasyCoder F4 (203 dpi)" /R=3 /P /DD

%END%

Around the Town,Jerome Davis,Buyer,1220 Governor Sq.,Hadden,
Alberta,WA1 1DP,Canada,(171) 555-7788,(171) 555-6750
```

The "`%BTW%`" is a Commander script command that tells Commander to launch BarTender and to use the string that follows as a BarTender command line. The "`%END%`" tells Commander that the command line is finished and what follows is data.

Of course, you must substitute in the command line, your own printer name, if it is different from **`EasyCoder F4 (203 dpi)`** and you must replace **`c:\command\`** with the path where you saved your *.btw files. (If there is a space anywhere in the path, you must enclose the entire value of the **`/AF=`** parameter in quotation marks.)

However, you should *not* replace `%Trigger File Name%` with invoice1.txt. Commander will recognize `%Trigger File Name%` as a variable on its own and it will automatically replace it with the latest trigger file name before it passes the parameter to BarTender. The `/R=3` tells BarTender to treat the third line in the trigger file as the data record. The `/P` tells BarTender to print the label and the `/DD` tells it to delete the trigger file after reading it.

2. Open the second sample database file, invoice2.txt, and add lines to the very beginning of the file so that the resulting file reads as follows.

```
%BTW% /AF=c:\command\SoftwareInvoice.btw /D=%Trigger File
Name% /PRN="Datamax 4400" /R=3 /P /DD

%END%

"Far Fetched Imports","Jerry Bom","manager","3320 Happy
St.","Incident","Washington","98887","USA","(444) 555-
7788","(444) 555-6750"
```

Note that this example assumes that the SoftwareInvoice.btw will be printed to a different printer than HardwareInvoice.btw. As in the previous sample trigger file, you should change the path and printer name shown here, as needed.

Start Detection

1. In Commander, select **Start Detection** from the **Detection** menu.

2. Copy the first sample file, invoice1.txt, back into the directory that Commander is monitoring.

<u>Commander will:</u>

- detect the file invoice1.txt,

- rename it to invoice1.old, and

- launch BarTender.

<u>Bartender will:</u>

- open HardwareInvoice.btw,

- read data from invoice1.old,

- print the label, and

- delete invoice1.old.

3. Copy the second sample file, invoice2.txt, back into the directory that Commander is monitoring.

Commander will:

- detect the file invoice2.txt,

- rename it to invoice2.old

- launch BarTender.

Bartender will:

- open SoftwareInvoice.btw,

- read data from invoice2.old,

- print the label, and

- delete invoice2.old.

Now you can have various invoice-creating applications automatically create files named *.txt in the directory that Commander is monitoring. The files should begin with a Commander script and data as in this example. Each time such a file is created in the directory, Bartender will print out the label format designated in the trigger file to the printer designated in the file.

Example 4: Print Label Using SAP R/3 IDOC File

Goal

The goal of this example is to show how a SAP™ (Systems, Applications, and Products in Data Processing) IDoc can be used as the trigger file and the data source.

SAP is an enterprise-scale, customizable, workflow application. SAP's success is built on its integration features, which enable disparate third-party applications and incompatible databases to exchange information with each other. BarTender uses one of these integration technologies, called Intermediate Documents (IDocs), to print data from any SAP-connected database onto labels. For more details about SAP and IDocs, refer to Seagull Scientific's white paper *Reading SAP IDocs*.

IDocs are hierarchically structured text files, and they can be used to trigger Commander, which, in turn, can instruct BarTender to use the IDoc as its data source.

Case Scenario

A company with an inventory management database running on a mainframe needs to print labels using BarTender running on Windows PCs. Using SAP R/3, the company creates a system that reads each new record in the database and creates an IDoc file containing the data of the record. Commander detects each new IDoc and uses BarTender to read the desired data from the IDoc and print a label.

To Implement this Scenario

Create a Sample IDoc

Note: The files created in this example can be found in 4thExample.zip inside CommanderExamples.zip located at:

ftp://ftp.seagullscientific.com/BarTender/Examples/Commander

Use them to verify that the files you create are correct.

1. Choose the network directory where the IDoc files will be saved. This is the directory that Commander will monitor. In this example, it will be called **\\Inventory\IDocs**.

2. Create, or obtain, a sample IDoc file of the same IDoc type as the IDocs that will be used for real data. Put it in the directory that you chose in step 1. In this example, we will assume that the IDoc type to be used is WTADDI01, and as the sample file we will use IDD-LIEF03_00108227.WTADDI0_46C, which is installed with BarTender.

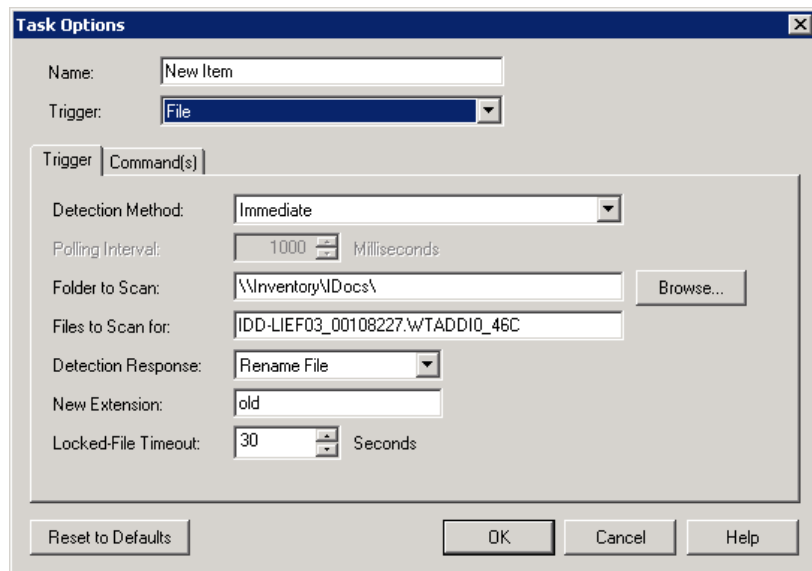3. Save the sample file in the directory that you chose in step 1.

Create the Label Format

1. In BarTender, select **New** on the **File** menu.

2. Select **Print** on the **File** menu and select the printer you want to use from the **Name** drop down list and then click **Close**.

3. Select **Page Setup** on the **File** menu and choose a stock or configure your page as needed.

4. Create a text object on the label and double-click it to open the **Modify Selected Text Object** dialog.

5. Click the **Data Source** tab and set the **Source** drop down list to **Database Field**.

6. Click the **Database Connection Setup** button.

7. In the **Add Database Connection Wizard**, click **Next**.

8. Select **SAP Intermediate Document (IDoc) File** and click **Next**.

9. Select the **IDoc Type Definition File** from the drop down list. In this example, the type is WTADDI01.

10. In the **IDoc File (Optional)** box, enter the complete path to the sample IDoc file. This is the same directory that you choose in step 1 of *Create a Sample IDoc*.

11. Click **Next**.

12. Select a **Master Segment**. (Click the **Help** button for details about Master Segments.) In this example, we select the line segment **E2WTADDI09000 (Additionals IDoc: Conditions)**.

13. Click **Next**.

14. At the **Fields** page of the wizard, click **Add Fields**.

15. Select the fields you would like and then click **OK**.

16. Click **Finish** on the **Add Database Connection Wizard**.

17. Click **OK** and you will be returned to the **Modify Selected Text Object** dialog with the **Data Source** tab open and the **Use Field** drop down list expanded. (If this does not happen automatically, click the **Data Source** tab and open the **Use Field** drop down list.)

18. Select a field to be the source for the text object, and then click **OK** to close the dialog.

19. Select **Save** in the **File** menu to save the label format under the name UsingSAPIDoc.btw.

20. Verify the connectivity to the IDoc file by printing the label.

21. Move the sample database file, IDD-LIEF03_00108227.WTADDI0_46C from the directory that Commander will be scanning.
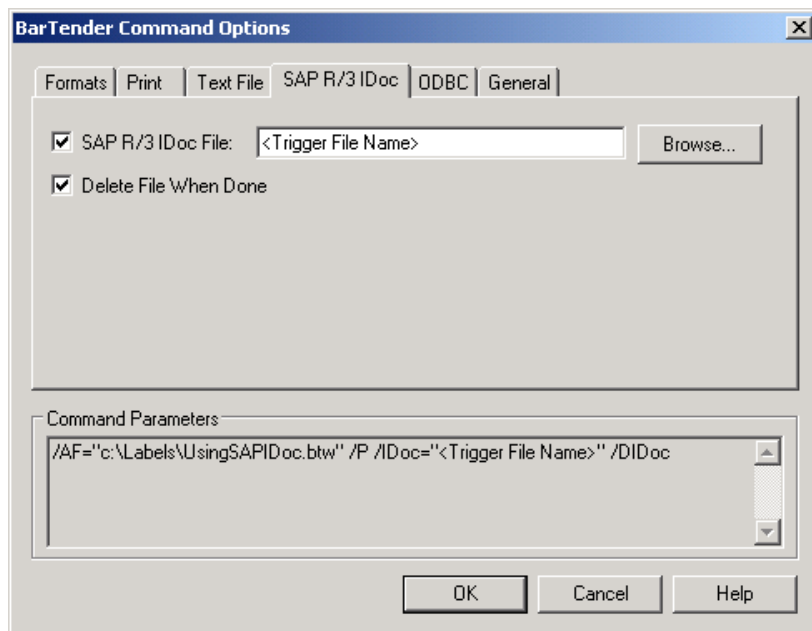
22. Close BarTender.

Create the Task

1. In Commander, select **New** on the **File** menu.

2. Select **Add** on the **Task** menu to open the **Task Options** dialog.

3. Enter "New Item" (without the quotation marks) as the task **Name** and select **File** in the **Trigger** drop down list.

4. Click the **Trigger** tab, and in the **Folders to Scan** box, enter the path of the directory that Commander will monitor. This is the same directory that you used in step 1 of *Create a Sample IDoc* above.

5. Enter "IDD-LIEF03_00108227.WTADDI0_46C" in the **Files to Scan for** box. (Do not include the quotation marks.)

6. Set the **Detection Response** to **Rename File**, rather than **Delete File** because BarTender cannot use the file as its database if Commander has deleted it.

7. Set the **New Extension** to "old" (without the quotation marks). This extension cannot be the same one specified in **Files to Scan for**.

8. Click the **Commands** tab and in the first row of the **Command Type** column, select **BarTender** from the drop down list.

9. Click the button labelled "**…**" at the end of the row to open the **BarTender Command Options** dialog. Note that there is initially just a **/P** parameter in the **Command Parameters** box. This tells BarTender to print.

10. Click the **Formats** tab and select the **File(s)** radio button. Then enter the path and file name for UsingSAPIDoc.btw. Note that an **/AF=** parameter has been added to the **Command Parameters** box identifying for BarTender which label format it should print.

11. Click the **SAP R/3 IDoc** tab and enable the **SAP R/3 IDoc File** check box, but leave the value at its default, the Commander variable %Trigger File Name%. An **/IDoc=%Trigger File Name%** parameter has been added to the **Command Parameters** box telling BarTender that it should use the trigger file as its database.

12. Enable the **Delete File When Done** checkbox to tell BarTender that it should delete the file after reading it. A **/DIDoc** parameter has been added to the **Command Parameters** box telling BarTender that it should delete the IDoc when it has finished reading data from it.

13. Click **OK** on the **BarTender Command Options** dialog.

14. Click **OK** on the **Task Options** dialog.

15. Select **Save** in the **File** menu and save the format under the name IDocReact.tl.

1. Select **Start Detection** from the **Detection** menu.

2. Copy the sample file, IDD-LIEF03_00108227.WTADDI0_46C, back into the directory that Commander is monitoring.

Commander will:

- detect the file IDD-LIEF03_00108227.WTADDI0_46C,

- rename it to IDD-LIEF03_00108227.old, and

- launch BarTender.

Bartender will:

- open UsingSAPIDoc.btw,

- read data from IDD-LIEF03_00108227.old,

- print the label, and

- delete IDD-LIEF03_00108227.old.

Available Seagull White Papers

General White Papers
- The Advantage of Drivers by Seagull

Companion Applications
- Printer Maestro: Enterprise Print Management
- Librarian
- BarTender Security Center
- BarTender Web Print Server

Recent Upgrades
- What's New in the Latest BarTender

Integration White Papers
- Integration Overview
- Commander
- Commander Examples
- BarTender's .NET SDKs
- BarTender's ActiveX Automation Interface
- Exporting Printer Code Templates
- Using BarTender with Terminal Services and Citrix MetaFrame
- Integration with Oracle's WMS and MSCA
- Integration with IBM WebSphere Sensor Events
- Integration with SAP

Miscellaneous White Papers
- Weighing Scales
- Dynamically Changing Objects at Print-Time using VB Script
- GHS Labeling
- Licensing for BarTender's Automation Editions
- Printing International Characters Using BarTender
- BarTender Software Activation
- Using BarTender's Application Identifier Wizard
- Optimizing Label Printing Performance
- Status Monitor Overview
- Silent Install

For downloadable versions, visit:
www.seagullscientific.com/aspx/white-papers.aspx

SEAGULL SCIENTIFIC

www.seagullscientific.com