BarTender®

The World's Leading Software for
Label, Barcode, RFID & Card Printing

White Paper

Commander

Middleware for Automatically Printing in Response to User-Defined Events

SEAGULL SCIENTIFIC

Contents

Overview of How Commander Works

Commander is a software utility, provided with both the Automation editions of BarTender, that enables you to perform automatic printing using BarTender in situations where it is not convenient or possible to perform automation using ActiveX or command lines. Commander can be run as an application or as a Windows service.

Triggers

When a system needs to print, it simply performs a triggering action, such as placing a file in a location of your choosing on the network, transmitting an e-mail, sending data via RS-232 (a serial port), or sending information through a TCP/IP socket connection. Commander detects the arrival or occurrence of these "triggers" and then "wakes up" BarTender so it can merge your data into the printed item design and automatically print your items.

Some Different Trigger Types

A trigger file or message can be an empty "wake up call" without data. Alternatively, the trigger can contain printed item data and script commands that will be used by Commander, BarTender, and/or another application that Commander launches.

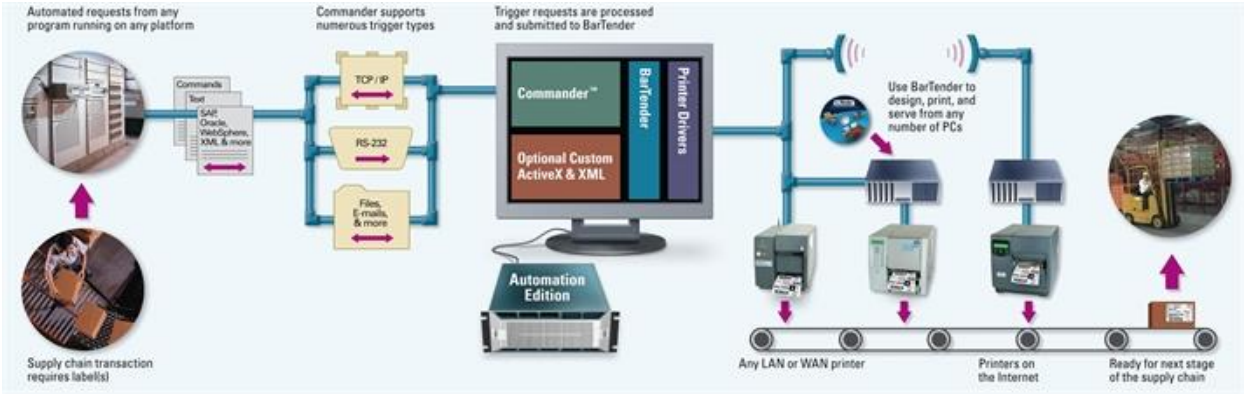
As an example, assume that an in-house order fulfillment program saves order information into a database and creates a trigger file named NewOrder.dat in a directory being watched by Commander. After Commander detects the creation of that file, it would "wake up" BarTender and the print job could be processed in a number of ways, including:

If the Trigger File is Empty: The BarTender document might be configured to query the database for information about orders entered after a specified time, and use the query results in the print job.

If the Trigger File Contains BarTender Document Data: Alternatively, the BarTender print job might be set up to simply read the document data right out of the trigger file.

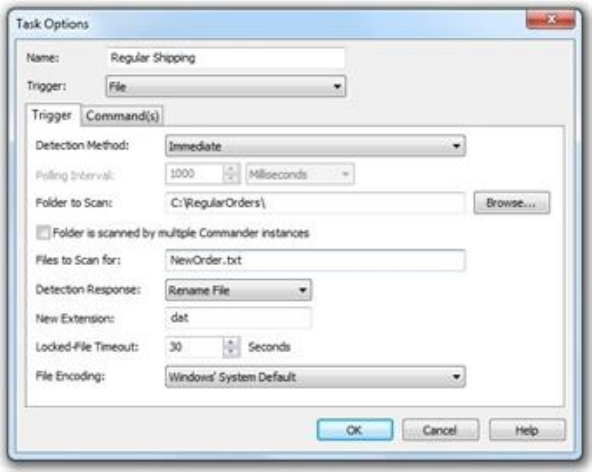
Other Types of Trigger Content: In addition to printed item data, trigger files and messages can also contain Commander Script and BarTender XML Script (BTXML).

Regardless of the trigger type, once the print job is done being processed, Commander would usually be configured to delete the NewOrder.dat trigger file so that it could resume monitoring of the directory and wait for the next print job.



Tasks

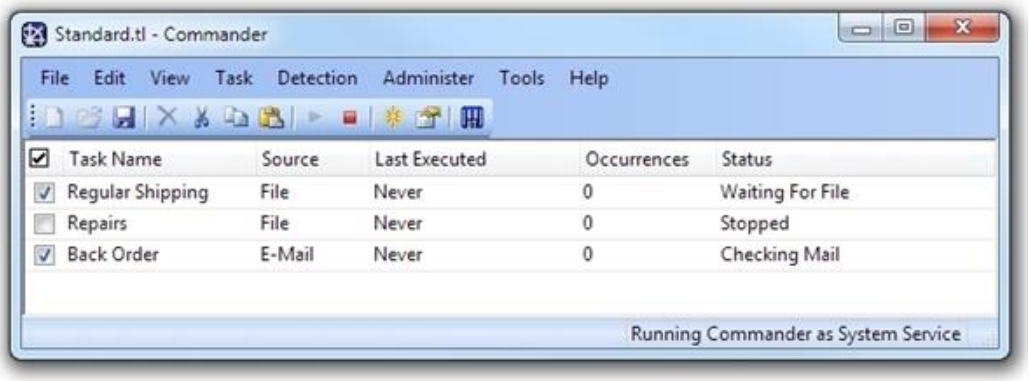
A Commander task is a set of one or more commands associated with a trigger.



- First you select what type of trigger will cause Commander to execute the command(s) for that task, and you specify certain options.
- Next you define the actual commands that will be executed in response to the trigger. These can be any combination of BarTender commands, operating system command lines, and Commander script.

Task Lists

Commander can monitor more than one trigger and execute a separate task in response to each one. The task or set of tasks that Commander is ready to execute at any given time is called a task list. You may define and save more than one task list, but only one can be active in Commander at any one time.



How Commander Detects Triggers

Commander's methods of detecting triggers are customizable. For example, Commander can detect file triggers using either of two methods:

- **Interrupt-driven:** Commander detects almost immediately that the trigger file has been created.
- **Polling-driven:** Commander checks the target directory at periodic intervals for the presence of the trigger file. You can set the interval with precision measured in milliseconds.

There are also multiple ways that TCP/IP Socket communications can trigger Commander, such as:

- Arrival of a specific sequence of characters.
- Occurrence of predefined periods of inactivity.
- Disconnection of the socket's source.

Using Commander to Run Any Application

Commander's primary purpose is to allow software programs to print items with BarTender without actually having to directly control BarTender. However, Commander can launch any system command or installed Windows application that can be executed using the **Run** option of the Windows **Start** menu. Therefore, Commander can respond to a received trigger by doing much more than just printing BarTender items.

For example, assume that a company's customer service department processes customers by having them come to web site to fill out forms requesting assistance and that the entered data is then transmitted to headquarters in the form of an e-mail. A copy of Commander running at headquarters could be configured to trigger when those emails arrive, write the e-mail contents to a text file, and then call an application that reads the file and stores its contents in a database. This is an example of how Commander's trigger-based integration technology can simplify your middleware challenges.

Advanced Commander Capabilities

Here are some more complex triggering scenarios that Commander can handle:

- It does not matter whether the application creating the trigger is running on Windows or not.
- The wildcard characters * and ? can be used to specify the trigger file name (or the Subject line when an e-mail trigger is used). This lets you easily take advantage of serialize file names (such as name001, name002, etc.) so that triggers can occur faster than Commander can process them.
- Special Commander variables can be used so that Commander can fill in certain values at task execution time and pass the data to BarTender.
- The contents of a trigger can contain the command line parameters that BarTender is to use when launched.
- The trigger contents can contain the data that should appear on a printed item.
- Commander can be configured to preload copies of BarTender and selected BarTender documents in advance. This way, the execution of tasks does not have to be slowed down by these loading actions, and can start printing right away.
- Commander can perform more than one command in response to a given trigger.
- Commander can instruct BarTender to close after it has printed the item.
- Commander can be set to log all of its activities.

Commander Scripts

A Commander script is a set of written instructions that Commander can read and execute. Applications that create triggers can add include script commands in the triggers. This allows the application to give Commander instructions that vary from trigger to trigger, instead of having Commander perform exactly the same actions in response to every trigger of a given type.

The primary purpose of a Commander script is to specify a BarTender command line, and to specify the data that is to populate the fields on the BarTender document.

Commander Variables

Commander Variables are text tags that get replaced at task execution time. They are used by different Command types in different ways. Some examples:

- With a **Commander Script** command, Commander Variables are replaced directly in the Commander Script.
- For an **Operating System** command, Commander Variables will be replaced in the Command Line.
- For a **BarTender** command, you can specify the name of a text file using a Commander Variable.

System environment variables can also be read using Commander Variables. This is useful for retrieving information like User Name or Computer Name. In the **Variables** tab of the **File > Task List Options** dialog, there is a table that allows you to create your own Commander Variables.

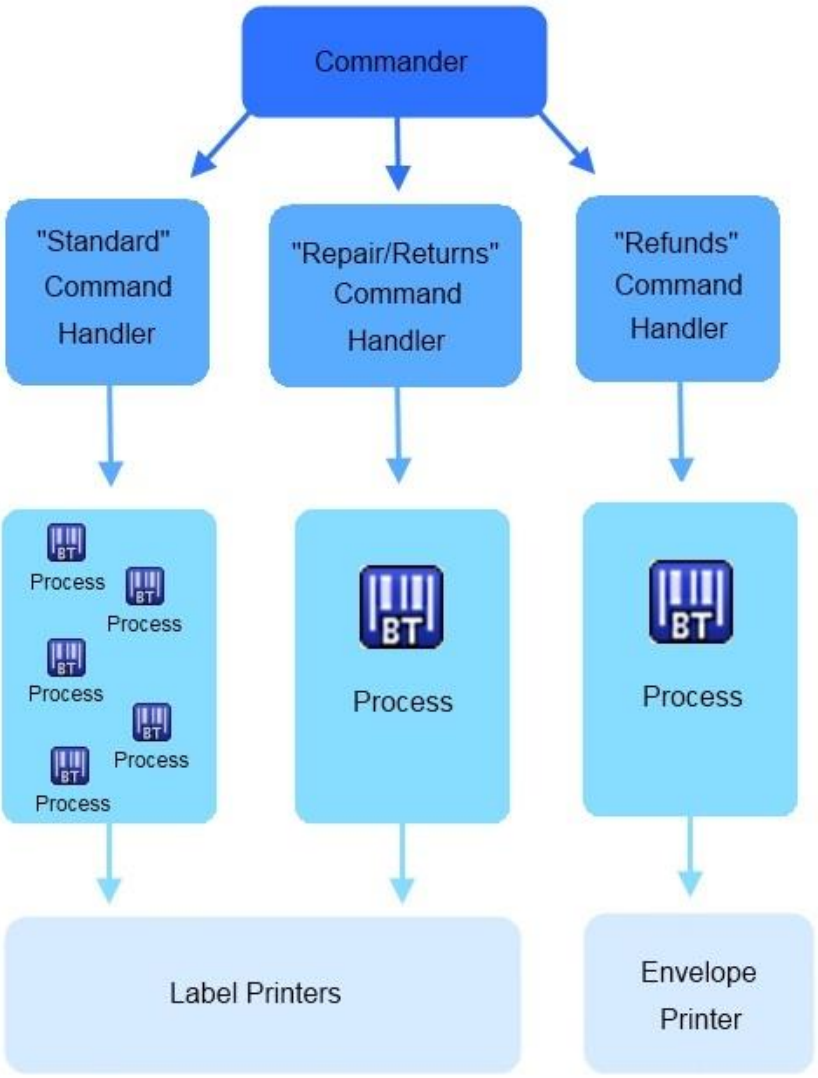
BarTender Command Handlers

You can increase the speed at which items are printed by having multiple instances (or “processes”) of BarTender running, each working simultaneously on the same or different tasks. Furthermore, different instances of BarTender can launch with different sets of command parameters, so you can assign each command of a task to the most appropriately-configured instance of BarTender. You can also save system resources by having seldom-triggered tasks immediately close BarTender after it is done being used to execute a task.

Each command in a task can specify a “Command Handler” to execute it. A Command Handler is a named group of settings that specify command line parameters to use when starting BarTender, as well as the number of BarTender processes to start. Command Handlers are primarily responsible for preloading Bartender documents, and passing print job instructions to BarTender processes. With the Enterprise Automation Edition (described below), if a particular Command handler is heavily used, you can define it to use more than one BarTender process. By default, all tasks share a single Command Handler. This minimizes the use of system memory, but does not perform as well in some cases if custom Command Handlers are used.

For example, a user could create these three command handlers:

- The command handler called “Standard” is used to create printed items for a high-volume shipping application. It preloads the same BarTender document into five different BarTender processes and leaves them running. Each time a new order is placed, a trigger is created by the shipping application and Commander uses the first available BarTender process to print the document. (The ability to assign multiple BarTender processes to a Command Handler is specific to the Enterprise Automation edition of Commander.)
- A second command handler, called “Repair/Returns,” is used less frequently. Therefore, it is sufficient for it to preload and use only one BarTender process.
- Finally, a third command handler, called "Refunds," is assigned to a task that is triggered to print an envelope only a few times a week. In order to conserve system resources, the Command Handler does not preload a BarTender process or any BarTender documents. Instead, it loads BarTender and the document as needed, prints the required envelope, and then closes BarTender.


Commander
"Standard" Command Handler
"Repair/Returns" Command Handler
"Refunds" Command Handler
Process
Process
Process
Process
Process
Process
Process
Label Printers
Envelope Printer

Commander Features added with Enterprise Automation Edition

TCP/IP Socket Triggers

In addition to triggering on Files and E-mails, the Enterprise Automation edition of Commander is capable of triggering on TCP/IP sockets. As with trigger files, TCP/IP socket communication can simply be an "empty" trigger, or the communication can also contain document data and/or Commander Script code. TCP/IP sockets are a good option when a file share or file creation is not possible or convenient. Direct TCP/IP socket communication is also faster than the creation of intermediary data files, and allows superior programmatic interaction between the originating application and Commander.

Ability to Submit BarTender XML Script (BTXML) and Receive Print Status Response

The Enterprise Automation edition of BarTender supports an XML-based scripting language that gives applications even more control of BarTender than Commander Script offers. Whereas Commander executes Commander Script directly and performs the requested actions itself, Commander passes on received BTXML for execution by BarTender.

Once BarTender executes the BTXML, it will return BTXML that contains detailed information about any executed print jobs, including the job status.

Send Data to a File, a TCP/IP Socket, or a Web Server

You can send trigger contents, the BTXML print response, or other data to a file, a socket, or a web server. This is particularly useful when returning print status.

XML Transforms

If XML is received by Commander as part of a trigger, Commander can use XSL transforms to convert it into any other data format. One of the best uses of this feature is to allow applications to output XML in their own familiar format and then depend on Commander to convert it into the XML script readable by BarTender (BTXML).

Search-and-Replace Transform

This transform allows you to automatically search and replace information from an incoming trigger. This is useful in a variety of circumstances, like changing a BarTender document name or a printer name.

Built-in Transforms for SAP AII, IBM WebSphere, and Oracle WMS and MSCA

These built-in transforms allow you to convert print request from these various systems into BTXML, so that they can be submitted to BarTender for processing.

Multiple BarTender Processes

With the Enterprise Automation edition of Commander, it is possible to assign multiple BarTender processes to a single Command Handler (described above, in the Bartender Command Handlers

section). This allows for more rapid response to large numbers of closely occurring Commander triggers, such as might be encountered in high-demand centralized printing environments.

Example 1: Print Item when Detecting New Trigger File

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 BarTender document. The techniques involved are also applicable to the later examples.

Case Scenario

At a parts-manufacturing plant there is a need to have an item with a serialized number printed at the end of an assembly line. On this item, there is no other dynamic or changing information, other than the incremented barcode. 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.

Implementation

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.

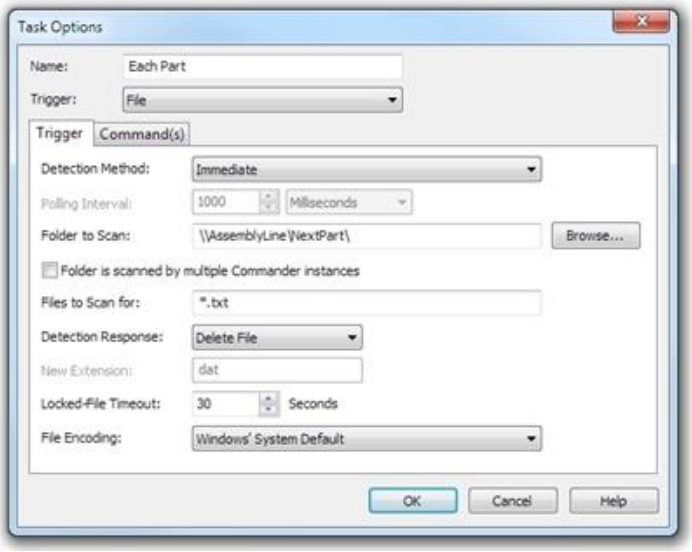
Create the BarTender Document

1. Open BarTender.
2. From the **File** menu, select **New**. The **New Document Wizard** opens.
3. Select **Blank Template**, then click **Finish** to close the **New Document Wizard**.
4. From the **File** menu, select **Print**. The **Print** dialog opens.
5. From the **Name** dropdown list, select the printer you want to use. Click **Close**.
6. From the **File** menu, select **Page Setup** and configure your page as needed. When you're finished, click **OK**.
7. From the **Create** menu, select **Barcode**.
8. From the **Select Barcode** dropdown list, select a barcode type, or select **More Barcodes** to open the **Select Barcode** dialog. (If you use the dialog, choose a barcode type, and click **Select**.)
9. Click anywhere on the template to add the barcode object.
10. Double-click the barcode. The **Barcode Properties** dialog opens.
11. Click the **Transforms** tab.
12. Next to **Serialization**, click the  icon. The **Serialization** dialog opens.
13. Select **Increment**.
14. Click **OK** to close the **Serialization** dialog.
15. Click **Close** to close the **Barcode Properties** dialog.

16. From the **File** menu, select **Save As**, and save the BarTender document under the name Serialized.btw.
17. Close BarTender.

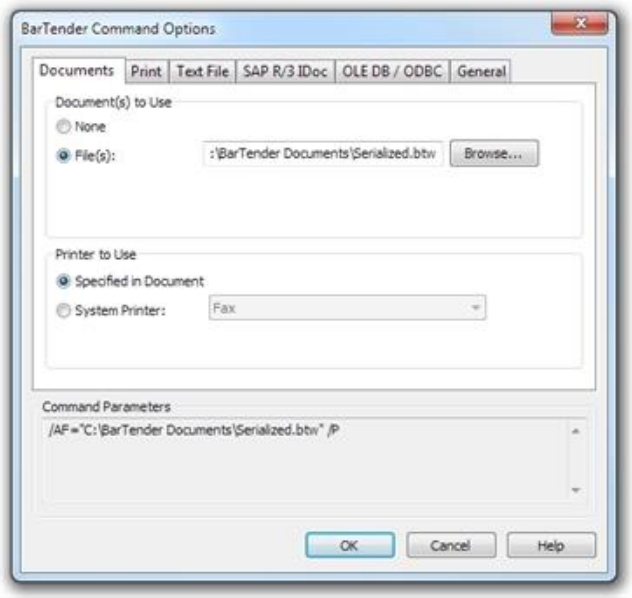
Create the Commander Task

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. Open Commander.
4. From the **File** menu, select **New**.
5. From the **Task** menu, select **Add** to open the **Task Options** dialog.
6. Enter "Each Part" (without the quotation marks) as the task Name, and select **File** in the **Trigger** drop down list.
7. Click the **Trigger** tab, and in the **Folder 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.
8. Enter "part.txt" or just "*.txt" in the **Files to Scan for** box. (Do not include the quotation marks.)
9. Set the **Detection Response** to **Delete File**, because BarTender will not need to open and read the file.



10. Click the **Commands** tab, and in the first row of the **Command Type** column, select **BarTender** from the dropdown list.
11. Click the button labeled "**...**" at the end of the row to open the **BarTender Command Options** dialog (opens on Documents tab). Note that there is initially just a `/P` parameter in the **Command Parameters** box. This tells BarTender to print.

12. Select the **Documents** tab, and select the **File(s)** radio button. Enter the path and file name for Serialized.btw. Note that an `/AF=` parameter has been added to the Command Parameters box, identifying which BarTender document to print.
13. Click **OK** to close the **BarTender Command Options** dialog, and then click **OK** to close the **Task Options** dialog.
14. In the **File** menu, select **Save As**, and save the task list under the name NewPart.tl.



Start Detection

1. From the **Detection** menu, select **Start Detection**.
2. Test your configuration by copying the sample file, part.txt, into the directory that Commander is monitoring.

Commander will:	BarTender will:
• detect the file part.txt • delete it • launch BarTender behind the scenes. (You won't see the BarTender application open on your desktop.)	• open Serialized.btw • print the item

With this setup, 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 item each time such a file is created.

Example 2: Print Item Using Trigger as Database

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 BarTender document at print-time.

Case Scenario

At a mail order company, there is a need to print an item 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 item.

Implementation

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.

Set Up a Sample Data File

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.,Haddonfield, 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 BarTender Document

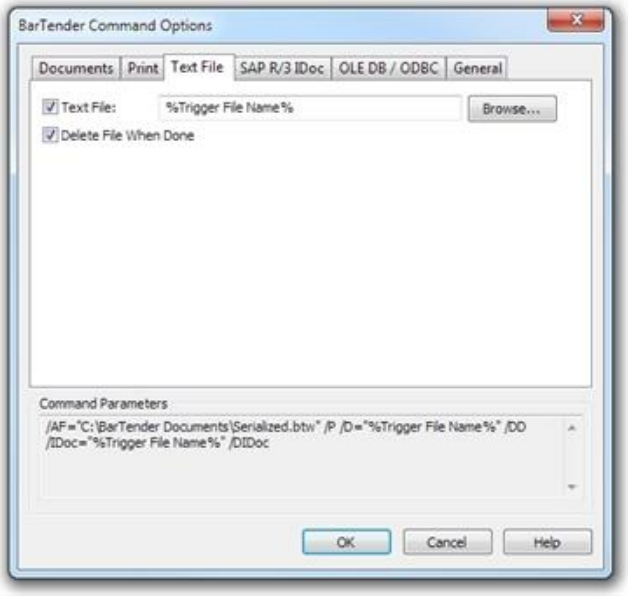
1. Open BarTender.
2. From the **File** menu, select **New**. The **New Document Wizard** opens.
3. Select **Blank Template**, then click **Finish** to close the **New Document Wizard**.
4. From the **File** menu, select **Print**. The **Print** dialog opens.
5. From the **Name** dropdown list, select the printer you want to use. Click **Close**.
6. From the **File** menu, select **Page Setup** and configure your page as needed. When you're finished, click **OK**.

7. From the **File** menu, select **Database Connection Setup**. The **Add Database Connection** wizard opens.
8. In the **Add Database Connection** wizard, click **Next**.
9. Select **Text File**, then click **Next**.
10. Enter the complete path and file name of the sample file you created, then click **Next**.
11. From the **Delimitation Type** dropdown list, select **Comma**, then click **Next**.
12. Select **No** when asked about the text file header.
13. Click **Finish** to close the wizard.
14. Click **OK** to close the **Database Connection Setup** dialog.
15. From the **Create** menu, select **Text**. From the **Text Objects** dropdown list, select a type of text object, and click on the template to add it.
16. Right-click the text object, and select **Properties**. The **Text Properties** dialog opens.
17. Click the **Data Source** tab, and click the Properties icon next to **Type**. The **Change Data Source Type Wizard** opens.
18. From the dropdown list, select **Database Field**. Click **Next**.
19. From the **Field Name** dropdown, select **Field 2** (because, in the sample, the second field contains the buyer's name).
20. In the **Sample Data** field, enter "buyer name" as the sample data, and click **Finish** to close the wizard.
21. Click **Close** to close the **Text Properties** dialog.
22. If other text objects are needed, simply return to step 15, in each case assigning the object to the appropriate field in the database. Otherwise, continue to step 23.
23. From the **File** menu, select **Save** to save the BarTender document under the name NewOrderAddress.btw.
24. Move the sample database file, orders.txt, out of the directory that Commander will be scanning.
25. Close BarTender.

Create the Commander Task

1. Open Commander.
2. From the Commander **File** menu, select **New**.
3. From the **Task** menu, select **Add** to open the **Task Options** dialog.
4. Enter "Each Order" (without the quotation marks) as the task Name.
5. In the **Trigger** dropdown list, select **File**.
6. Click the **Trigger** tab.
7. In the **Folder 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.
8. Enter "order.txt" or just "*.txt" in the **Files to Scan for** box (without the quotation marks).

9. Set the **Detection Response** to **Rename File**, since BarTender cannot use the file as its database if Commander has deleted it.
10. Set the **New Extension** to "old" (without the quotation marks). This extension cannot be the same one specified in **Files to Scan for**.
11. Click the **Commands** tab, and in the first row of the **Command Type** column, select **BarTender** from the dropdown list.
12. 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.
13. Click the **Documents** 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 which BarTender document to print.
14. Click the **Text File** tab and enable the **Text File** check box, but leave the value at its default, %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.
15. 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.
16. Click **OK** to close the **BarTender Command Options** dialog.
17. Click **OK** to close the **Task Options** dialog.



18. From the **File** menu, select **Save**, and save the task list under the name NewOrderAddress.tl.

Start Detection

1. From the **Detection** menu, select **Start Detection**.
2. Test your setup by copying the sample file, orders.txt, back into the directory that Commander is monitoring.

Commander will:	BarTender will:
• detect the file orders.txt • rename it to orders.old • launch BarTender behind the	• open NewOrderAddress.btw • read data from orders.old • print the item

scenes. (You won't see the BarTender application open on your desktop.)	• delete orders.old

With this setup, 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 an item for each record in the file.

Example 3: Print Item Using Commander Script

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 printing needs, scattered throughout their network. In such cases, the BarTender document 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 that are 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.

Implementation

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.

Set Up Sample Data Files

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.,Haddonfield, 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 BarTender Documents

1. Following the same steps as the Create the BarTender Document section of the previous example, create a BarTender document for the first department, using `invoice1.txt`, created above, as the database. Use `HardwareInvoice.btw` as the document's file name.
2. Create a second BarTender document 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 BarTender document'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. Open Commander.
2. From the Commander **File** menu, select **New**.
3. From the **Task** menu, select **Add** to open the **Task Options** dialog.
4. Enter "Invoices" (without the quotation marks) as the task Name.
5. From the **Trigger** dropdown list, select **File**.
6. Click the **Trigger** tab.
7. In the **Folder 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.
8. Enter "*.txt" in the **Files to Scan for** box. (Do not include the quotation marks.)
9. Set the **Detection Response** to **Rename File**, since BarTender cannot use a file as a database if Commander has deleted it.
10. Set the **New Extension** to "old" (without the quotation marks). This extension cannot be the same one specified in **Files to Scan for**.

11. Click the **Commands** tab, and in the first row of the **Command Type** column, select **Commander Script** from the dropdown list.
12. Click **OK** to close the **Task Options** dialog.
13. From the **File** menu, select **Save**, and save the file 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 `%END%`.)

```
%BTW% /AF=c:\command\HardwareInvoice.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.,Haddonfield, Alberta,WA1
1DP,Canada,(171) 555-7788,(171) 555-6750
```

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

   Of course, you must substitute your own printer name in the Commander Script, if it is different from `EasyCoder F4 (203 dpi)` and you must replace `c:\command\` with the path where you saved your *.btw files.

   > **NOTE:** If any of the paths, file names, or printer names used in your Commander Script contain spaces, you must enclose the entire path in quotes.

   Do *not* replace `%Trigger File Name%` with the name of the database file. 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` command tells BarTender to treat the third line in the trigger file as the data record, and the `/P` command tells BarTender to print the item. Finally, `/DD` tells BarTender 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. In both trigger files, you should change the path and printer name following the `/PRN` command, as needed.

Start Detection

1. From the **Detection** menu, select **Start Detection**.
2. Copy the first sample file, invoice1.txt, back into the directory that Commander is monitoring.

Commander will:	BarTender will:
• detect the file invoice1.txt • rename it to invoice1.old • launch BarTender behind the scenes. (You won't see the BarTender application open on your desktop.)	• open HardwareInvoice.btw • read data from invoice1.old • print the item • delete invoice1.old

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

Commander will:	BarTender will:
• detect the file invoice2.txt • rename it to invoice2.old • launch BarTender behind the scenes. (You won't see the BarTender application open on your desktop.)	• open SoftwareInvoice.btw • read data from invoice2.old • print the item • 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 document designated in the trigger file to the printer designated in the file.

Example 4: Print Item 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 print items.

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 items 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 the item.

Implementation

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.

Create a Sample IDoc

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`. 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 BarTender Document

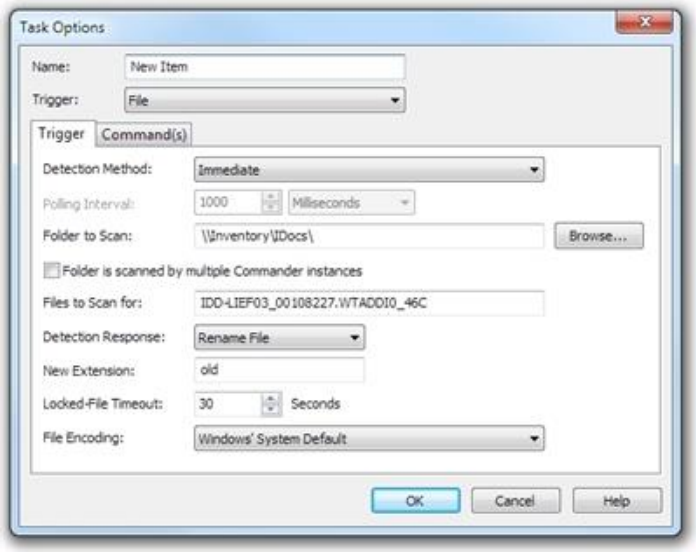
1. Open BarTender.
2. From the **File** menu, select **New**. The **New Document Wizard** opens.
3. Select **Blank Template**, then click **Finish** to close the **New Document Wizard**.
4. From the **File** menu, select **Print**. The **Print** dialog opens.
5. From the **Name** dropdown list, select the printer you want to use. Click **Close**.

6. From the **File** menu, select **Page Setup** and configure your page as needed. When you're finished, click **OK**.
7. From the **File** menu, select **Database Connection Setup**. The **Add Database Connection** wizard opens.
8. In the **Add Database Connection** wizard, click **Next**.
9. Select **SAP Intermediate Document (IDoc) File** and click **Next**.
10. Select the **IDoc Type Definition File** from the dropdown list. In this example, the type is WTADDI01.
11. 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.
12. Click **Next**.
13. Select a Master Segment. In this example, we select the line segment **E2WTADDI09000 (Additionals IDoc: Conditions)**.
14. Click **Next**.
15. On the **Fields** page of the wizard, click **Add Fields**.
16. Select the fields you would like, and then click **OK**.
17. Click **Finish** on the **Database Connection Setup** dialog.
18. Click **OK** to close the **Add Database Connection** wizard
19. From the **Create** menu, select **Text**. From the **Text Objects** dropdown list, select a type of text object, and click on the template to add it.
20. Right-click the text object, and select **Properties**. The **Text Properties** dialog opens.
21. Click the **Data Source** tab, and click the Properties icon next to **Type**. The **Change Data Source Type Wizard** opens.
22. From the dropdown list, select **Database Field**. Click **Next**.
23. Select a field to be the source for the text object, and then click **Finish** to close the wizard.
24. Click **Close** to close the **Text Properties** dialog.
25. From the **File** menu, select **Save** to save the BarTender document under the name UsingSAPIDoc.btw.
26. Verify the connectivity to the IDoc file by printing the item.
27. Move the sample database file `IDD-LIEF03_00108227.WTADDI0_46C` out of the directory that Commander will be scanning.
28. Close BarTender.

Create the Task

1. Open Commander.
2. From the Commander **File** menu, select **New**.
3. From the **Task** menu, select **Add** to open the **Task Options** dialog.

4. Enter "New Item" (without the quotation marks) as the task Name.
5. In the **Trigger** dropdown list, select **File**.
6. Click the **Trigger** tab.
7. In the **Folder 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.
8. Enter “IDD-LIEF03_ 00108227.WTADDI0_46C" in the **Files to Scan for** box (without the quotation marks).



9. Set the **Detection Response** to **Rename File**, since BarTender cannot use the file as its database if Commander has deleted it.
10. Set the **New Extension** to "old" (without the quotation marks). This extension cannot be the same one specified in **Files to Scan for**.
11. Click the **Commands** tab, and in the first row of the **Command Type** column, select **BarTender** from the dropdown list.
12. 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.
13. Click the **Documents** 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 BarTender document it should print.
14. Click the **SAP R/3 IDoc** tab and enable the **SAP R/3 IDoc File** check box, but leave the value at its default, %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.
15. 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.
16. Click **OK** to close the **BarTender Command Options** dialog.
17. Click **OK** to close the **Task Options** dialog.
18. From the **File** menu, select **Save**, and save the task list under the name IDocReact.tl.

Start Detection

1. From the **Detection** menu, select **Start Detection**.
2. Copy the sample file, IDD-LIEF03_00108227.WTADDI0_46C, back into the directory that Commander is monitoring.

Commander will:	BarTender will:
• detect the file IDD-LIEF03_ 00108227.WTADDI0_46C • rename it to IDD-LIEF03_ 00108227.old • launch BarTender behind the scenes. (You won't see the BarTender application open on your desktop.)	• open UsingSAPIDoc.btw • read data from IDD-LIEF03_ 00108227.old • print the item • delete IDD-LIEF03_00108227.old

Related Documentation

White Papers

General White Papers

- The Advantage of Drivers by Seagull
- Licensing for BarTender's Automation Editions
- Silent Install

Companion Applications

- Printer Maestro: True Enterprise Print Management for Windows
- Librarian
- BarTender Security Center
- BarTender Web Print Server

Recent Upgrades

- What's New in the Latest BarTender

For downloadable versions, visit:

www.seagullscientific.com/label-software/white-papers.aspx