3

Datacap Forms

This chapter begins with a brief introduction to the **Datacap forms** that provide a task with dialogs for ***setup*** criteria, and for ***runtime*** procedures that carry out a task's operations.

Chapter 3 shows you how to identify, assess and modify the inventory of **stock** forms – and how to assign **stock** forms to a Task Project. This chapter also explores the steps you take to assemble, test and implement new Datacap forms.

Chapter 3 covers these topics:

I. Introduction 3-2
- **A. Task Projects - Setup Forms and Runtime Forms 3-3**
- **B. Multi-tab Forms – an Introduction 3-6**
- **C. Data Entry Panels 3-7**

II. Controls of Datacap Forms 3-9
- **A. How to Add Controls to the Development Toolbox 3-11**

III. Scripts of the Datacap Form 3-13

IV. Developing Datacap Forms 3-14
- **A. Setup Forms and Dialogs 3-18**
  - i. Controls of a Setup Form 3-19
  - ii. How to Construct a Setup Form 3-23
- **B. Runtime Forms and Dialogs 3-26**
  - i. How to Construct a Runtime Form 3-29

V. How to Test a Form 3-33

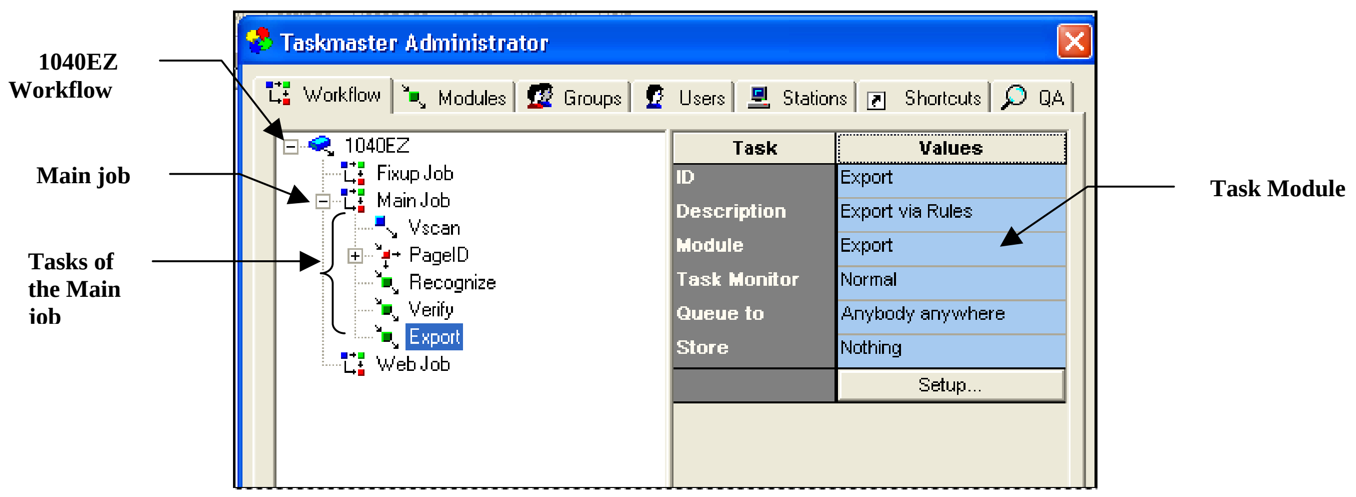
VI. Multi-tab Forms 3-35

VII. Images 3-37

VIII. Hot Keys 3-39
- **A. How to Define a Hot Key 3-40**

Introduction

The *1040EZ* training application has a Main job with five tasks:



Task	Values
ID	Export
Description	Export via Rules
Module	Export
Task Monitor	Normal
Queue to	Anybody anywhere
Store	Nothing

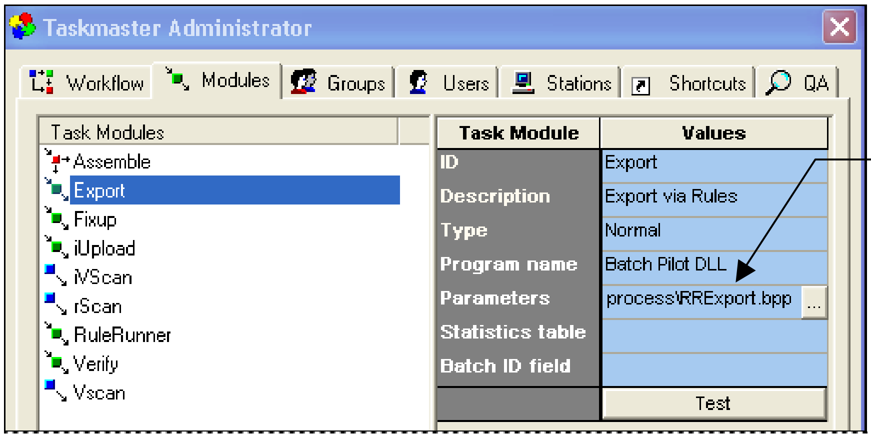
1040EZ Taskmaster Administrator – Workflow tab

✓ This ***pre-configured*** application is easy to access and explore – and to experiment with. To access and examine *1040EZ*:

Step	Action
1.	Select **Datacap Taskmaster** from the **Programs** options of your computer's Windows Start button.
2.	Open the **Applications** folder and the **1040EZ** sub-folder.
3.	Click on **1040EZ Client**.
4.	Enter your administrator's User ID and Password in the ***Login*** dialog; press the OK button.
5.	When the ***1040EZ Taskmaster Window*** opens, close the secondary ***Operations*** window.
6.	Select **Workflow** from the **Settings** menu to open the *Workflow* tab of the *1040EZ* application's ***Taskmaster Administrator*** (pictured above).
7.	In the **Components** area on the left, open the *Main Job* and highlight the *Export* Task ID.
8.	Take a moment to review the task's **Properties** on the right, noting especially the name of the task's **Task Module** – *Export*, in this case.
9.	Move to the ***Taskmaster Administrator's*** *Modules* tab (illustrated on the next page.)

To Access and Work with the 1040EZ Application (continued)

Step	Action
10.	Highlight the Export module and look carefully at its **Parameters** value: `\Datacap\1040EZ\process\RRExport.bpp` is the name and path of the **Task Project** that is the foundation for the Export task.



Task Module	Values
ID	Export
Description	Export via Rules
Type	Normal
Program name	Batch Pilot DLL
Parameters	process\RRExport.bpp
Statistics table	
Batch ID field	

1040EZ Taskmaster Administrator – *Modules tab*

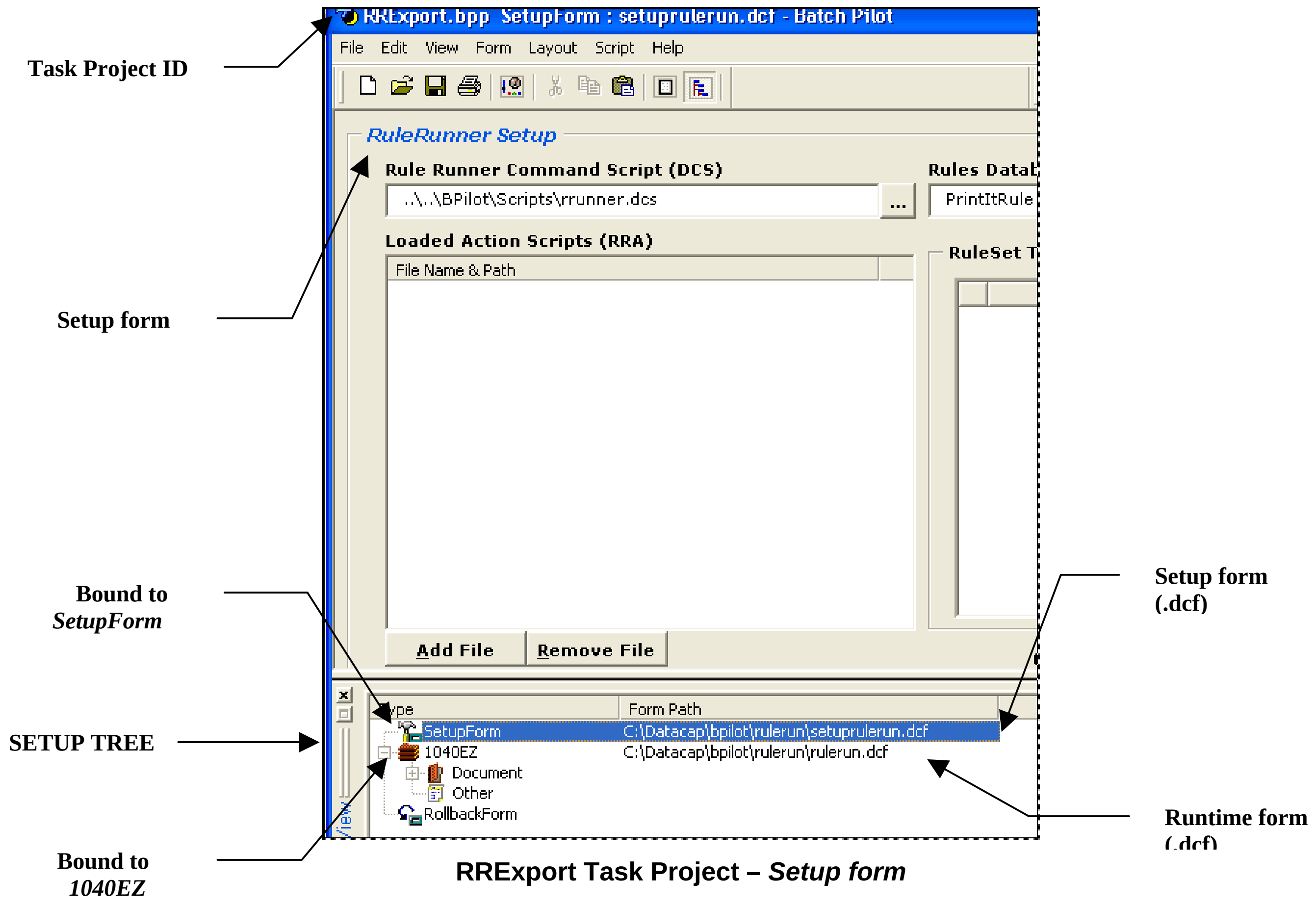
☛ Like most Task Projects, **Export.bpp** employs two **Datacap forms (.dcf)**:

- An Administrator uses the dialog that is based on the ***setup*** form to assemble a Task Definition;
- The task uses a dialog based on the ***runtime*** form to track and display its operating progress. In the case of a Verify task, the form's ***Data Entry*** panel gathers important data and additional settings.

Task Projects - Setup Forms and Runtime Forms

For a first-hand look at the Export Task Project (.bpp) and its two Datacap forms (.dcf):

Step	Action
1.	Open *Batch Pilot* and the ***Batch Pilot Window***. (Chapter 2 is a complete guide to the ***Batch Pilot Window***.)
2.	Select **Open Project** from the **File** menu.
3.	Use the ***Open Files*** dialog to access the **1040EZ** folder of your **Datacap** directory.
4.	Open the **Process** sub-folder and select the **RRExport.bpp** Task Project file.
5.	When the project and its ***setup*** form appear in the window, un-select the **Form** menu's **Design** item for a clearer look at the project and the form (illustrated on the next page).


Task Project ID
Setup form
Bound to SetupForm
SETUP TREE
Bound to 1040EZ
Setup form (.dcf)
Runtime form (.dcf)
RRExport.bpp SetupForm : setuprulerun.dcf - Batch Pilot
File Edit View Form Layout Script Help
RuleRunner Setup
Rule Runner Command Script (DCS)
..\..\BPilot\Scripts\rrunner.dcs
Rules Datab
PrintItRule
Loaded Action Scripts (RRA)
File Name & Path
RuleSet T
Add File
Remove File
Type
Form Path
SetupForm
C:\Datacap\bpilot\rulerun\setuprulerun.dcf
1040EZ
C:\Datacap\bpilot\rulerun\rulerun.dcf
Document
Other
RollbackForm
View

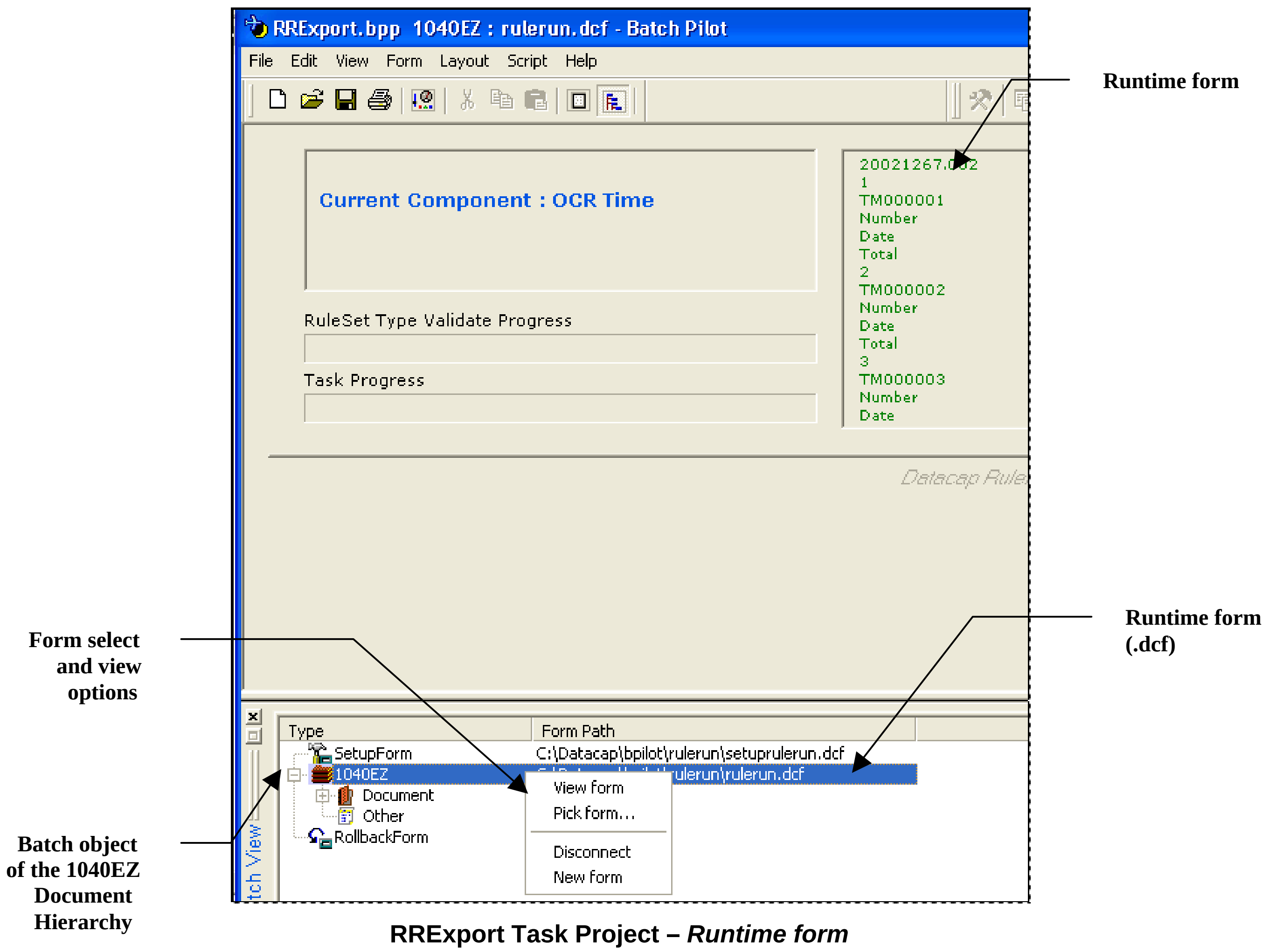
RRExport Task Project – *Setup form*

In the illustration above:

- The Title Bar identifies the Task Project (**RRExport.bpp**) and the ***setup*** form that is currently open in the window's Data Area: **setuprulerun.dcf**.
- The Data Area displays a portion of the ***setup*** form as part of the Task Project.
- The **Setup Tree** in the window's **Batch View** area in the lower left-hand corner "binds" the ***setup*** form to the *SetupForm* item in the **Type** column: the **Form Path** column identifies the name and path of the **RRexport.bpp** Task Project's ***setup*** form (.dcf). This means that the form will serve as a task's ***setup*** dialog during the Task Definition process.
- The **Setup Tree** also binds a ***runtime*** form (**rulerun.dcf**) to the *1040EZ* **Batch** object of the application's Document Hierarchy. As a result, the *1040EZ* task that is based on this Task Project will use the form as its operations dialog when it processes a batch and its contents.

✓ Pay close attention to the forms' names and paths. These are **stock** forms that are usually assigned to the Task Projects for an application's *RuleRunner* tasks. These forms, and other **stock** forms, reside in sub-folders of your **Datacap** directory's **BPilot** folder.

When they run, tasks of the *RuleRunner* type apply rules that you have defined; you use the ***setup*** form during the Task Definition process to identify those rules (Chapter 4). For explanations of *RuleRunner* tasks, see Chapter 8 of the *Guide to Taskmaster Rules*.


Runtime form
Runtime form (.dcf)
Form select and view options
Batch object of the 1040EZ Document Hierarchy
RRExport.bpp 1040EZ : rulerun.dcf - Batch Pilot
File Edit View Form Layout Script Help
Datacap Rule
View form
Pick form...
Disconnect
New form

Current Component : OCR Time

RuleSet Type Validate Progress

Task Progress

```
20021267.002
1
TM000001
Number
Date
Total
2
TM000002
Number
Date
Total
3
TM000003
Number
Date
```

Type	Form Path
SetupForm	C:\Datacap\bpilot\rulerun\setuprulerun.dcf
1040EZ	…rulerun\rulerun.dcf
Document	
Other	
RollbackForm	

RRExport Task Project – *Runtime form*

Because the *1040EZ* application is already fully configured – and the Export task's Task Project (**RRExport.bpp**) has been defined – the project's ***runtime*** form has been assigned to the *1040EZ* **Batch** object of the application's Document Hierarchy.

Again, this is a **stock** form, with the following name and path:

```
C:\Datacap\bpilot/rulerun\rulerun.dcf
```

✓ To view the form, right-click on the applicable row in the **Batch View** area, and select **View form** from the options.

This chapter explores the techniques you use to assemble new Datacap forms and manage existing forms. Before proceeding, take time to investigate the forms of other *1040EZ* projects. Sample, for example, the forms of the *1040EZ* application's FixUp Task Project.

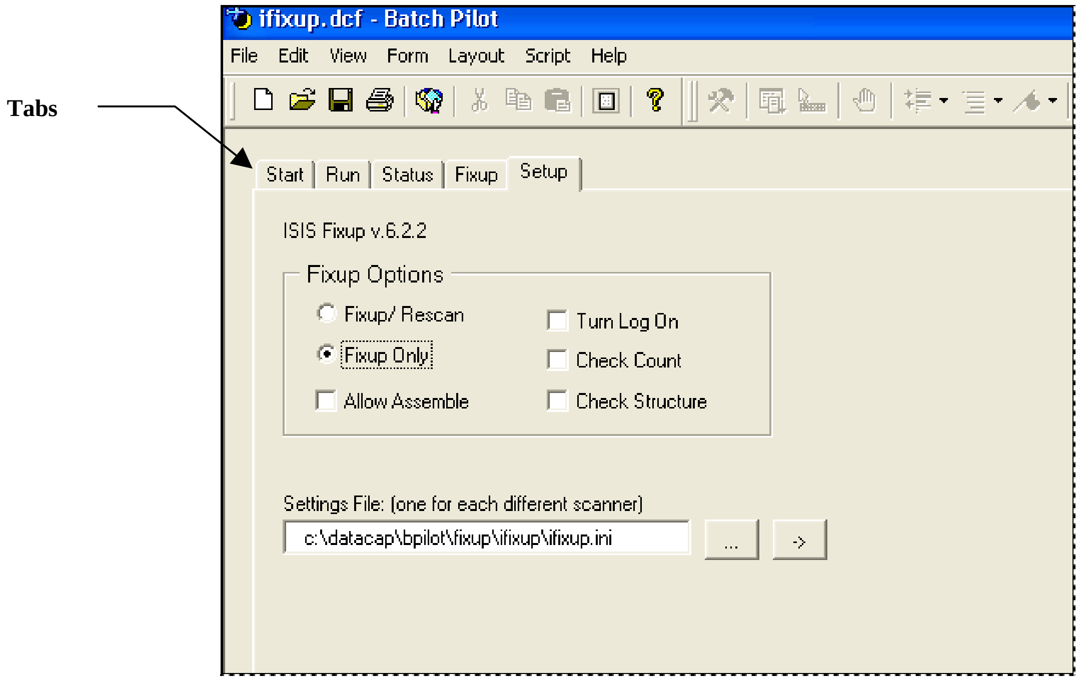
You'll find **FixUp.bpp** in the application's **Process** directory. Try, too, the forms of **verify.bpp** - the application's Verify Task Project . Although this project's ***setup*** form is unremarkable, the ***runtime*** form is a complete ***Data Entry*** panel (Chapter 5).

Multi-tab Forms – an Introduction

Some **stock** forms consist of multiple tabs: each will end up as a dialog responsible for a specific portion of a task's ***setup*** or ***runtime*** procedures.

To look at one example, use the tools of the ***Batch Pilot Window*** to locate and open the **stock ifixup.dcf** form – which you will find in this location: **Datacap/BPilot/FixUp/iFixUp.**

This form has five tabs: *Setup* is this Multi-tab Form's ***setup*** component; *FixUp* is its principal ***runtime*** component and is supported by the remaining tabs.



ISIS Fixup v.6.2.2

Fixup Options

( ) Fixup/ Rescan
(•) Fixup Only
[ ] Allow Assemble
[ ] Turn Log On
[ ] Check Count
[ ] Check Structure

Settings File: (one for each different scanner)

c:\datacap\bpilot\fixup\ifixup\ifixup.ini

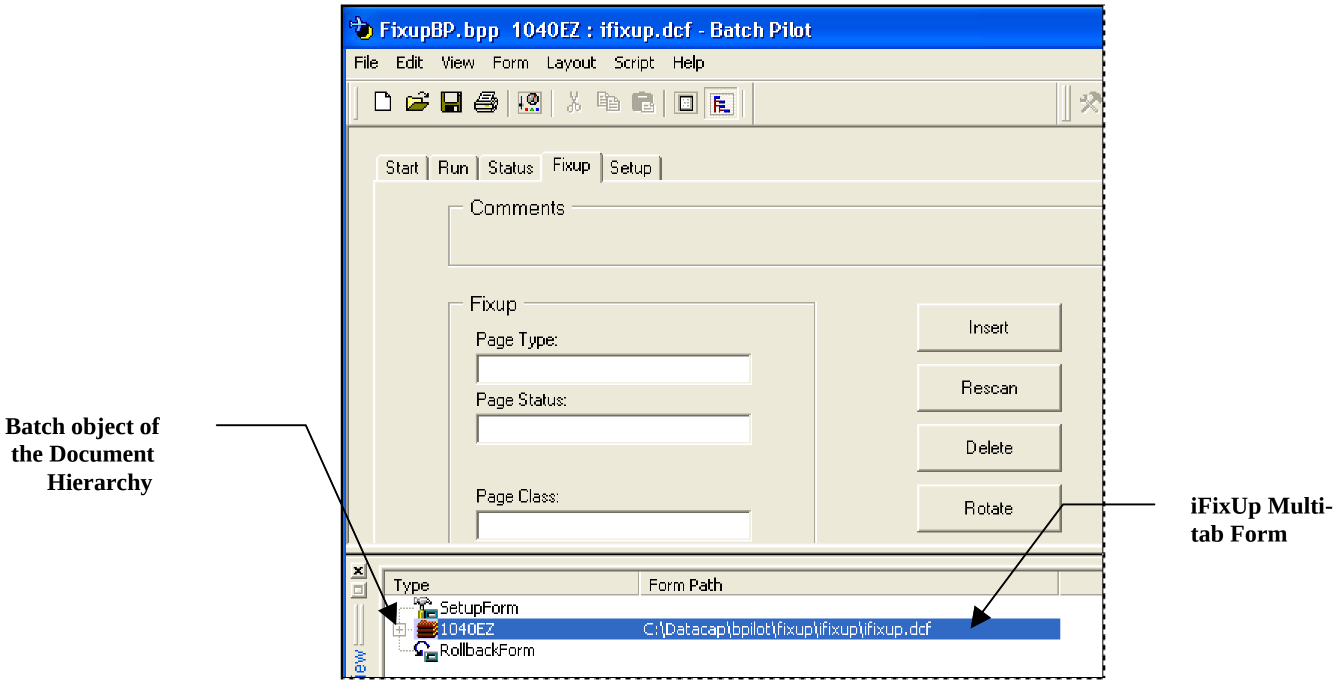
iFixUp Multi-tab Form – *Setup tab*

If you open a Task Project that employs this Multi-tab Form – the *1040EZ* application's **FixUpBP.bpp** project, for example – two features are apparent (after you de-select the **Form** menu's **Design** item!):

- The Multi-tab Form itself is linked to the **Batch** object of the application's Document Hierarchy (*1040EZ*) rather than to the *SetupForm* **Type** (see the illustration on the next page.)
- You can use your cursor to move freely from tab to tab, for a better look at the contents of each.

✓ Although this form is not formally designated a *Setup Form*, ***only*** the contents of the *Setup* tab appear as a dialog during the Task Setup process. The section that begins on

Page 33 examines this important distinction, and shows you how to construct a Multi-tab Form, with multiple tabs.


FixupBP.bpp 1040EZ : ifixup.dcf - Batch Pilot
File Edit View Form Layout Script Help
Start Run Status Fixup Setup
Comments
Fixup
Page Type:
Page Status:
Page Class:
Insert
Rescan
Delete
Rotate
Type
Form Path
SetupForm
1040EZ
C:\Datacap\bpilot\fixup\ifixup\ifixup.dcf
RollbackForm
Batch object of the Document Hierarchy
iFixUp Multi-tab Form

iFixUp Multi-tab Form – *Fixup tab*

Data Entry Panels

A ***Data Entry*** panel is not a **stock** form because its contents are application-specific. Each field in the panel is a **Field** object of the application's Document Hierarchy, and the pages that contain these fields are unique **Page** objects as well. Therefore, the panel of one application is substantially different than the ***Data Entry*** panel of any other application.

✓ There is another difference: the ***runtime*** form of an application's Verify Task Project is usually bound to a **Page** object of the Document Hierarchy (see the illustration on the next page.)

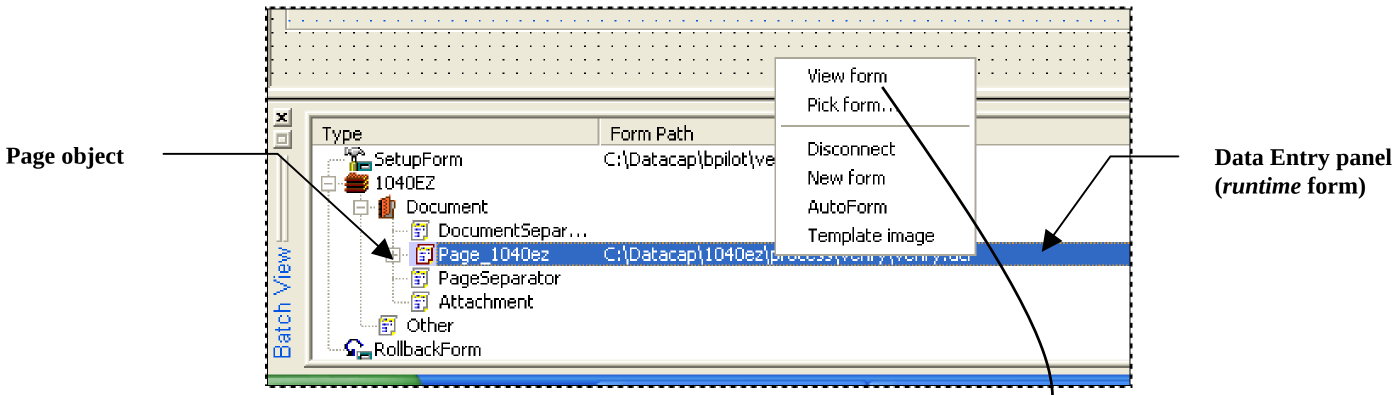
Datacap Taskmaster's *Autoform* utility refers to the Document Hierarchy as it constructs a ***Data Entry*** panel. Chapter 5 explains this technology as it shows you how to assemble, test and modify this essential form.

✓ Two procedures give you access to the structure of the *1040EZ* application's ***Data Entry*** panel (as an example):

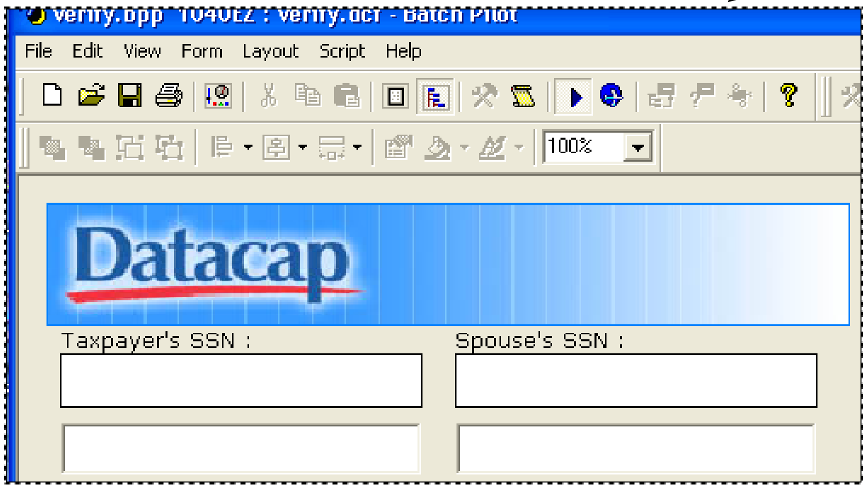
- Using the tools of the ***Batch Pilot Window***, you can open the application's **verify.dcf** form – located in the **Datacap** directory's **1040EZ** folder…in the **Verify** sub-folder of the **Process** directory.

- Instead, you can access the **Process** directory and open the **verify.bpp** Task Project (rather than the form itself):

    1. In the **Batch View** area, open the *1040EZ* Document Hierarchy, until you can see the *Page_1040ez* **Page** object – and the ***runtime*** form's name and path in the **Form Path** column.
    2. Right-click on this row and select the **View Form** option.



Verify.bpp Task Project
<u>Batch View area</u>



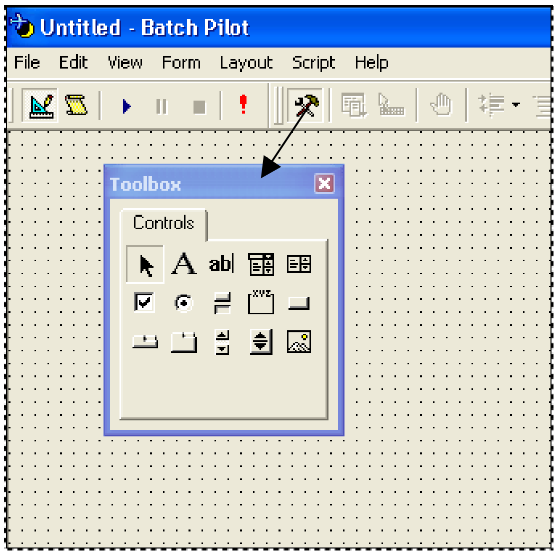
1040EZ Data Entry Panel – upper fields (left)

✓ ***Very Important!*** Keep in mind that the form for the *1040EZ* ***Data Entry*** panel is already in place – and has been previously bound to the *Page_1040EZ* **Page** object of the application's Document Hierarchy.

Controls of Datacap Forms

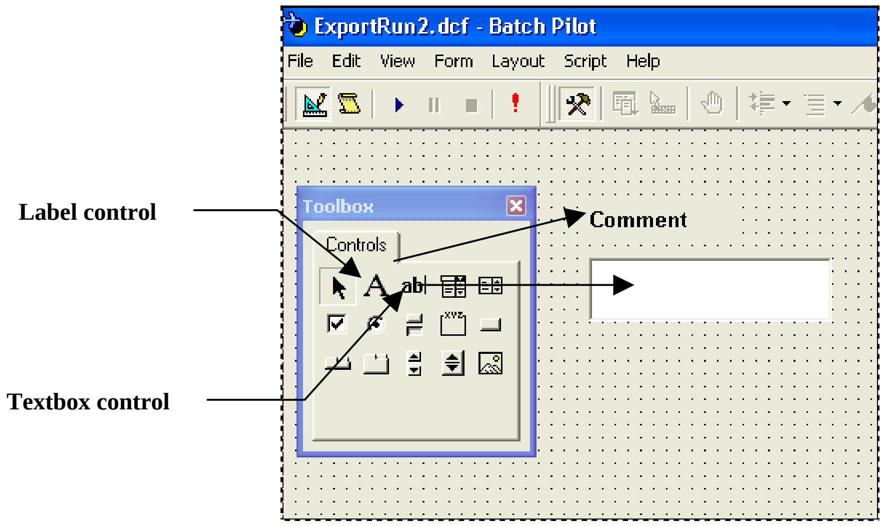
The **Development Toolbox** of the ***Batch Pilot Window*** (Chapter 2) has a small set of controls that are easy to access and work with when you are developing a form – and just as friendly to the form's user.

To access the toolbox, toggle the **Options** icon on the **Script** toolbar, or select **Toolbox** from the **Form** menu.



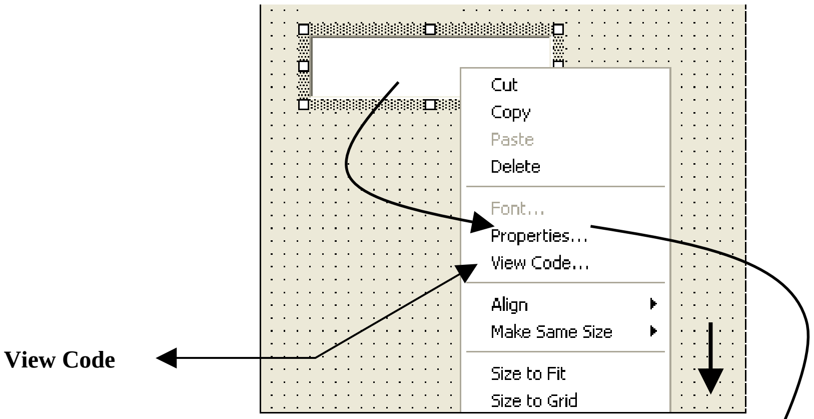
Batch Pilot Window – *Design Mode*

✓ Below, the developer has dragged a **Label** and **Textbox** control onto a new form.

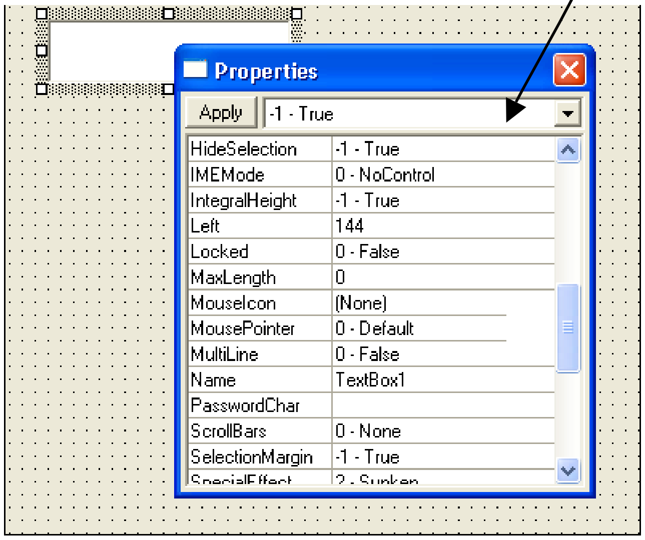

Label control
Textbox control
ExportRun2.dcf - Batch Pilot
File Edit View Form Layout Script Help
Toolbox
Controls
Comment

Label and Textbox Controls

Right-clicking on a control reveals a list of options for management of the control; selecting **Properties** displays the control's ***Properties*** dialog (illustrated on the next page.)



Options – Textbox Control

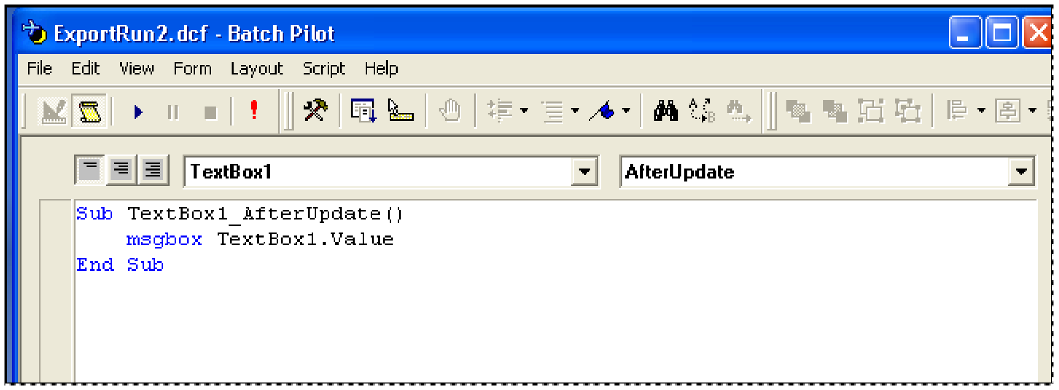


-1 - True

HideSelection	-1 - True
IMEMode	0 - NoControl
IntegralHeight	-1 - True
Left	144
Locked	0 - False
MaxLength	0
MouseIcon	(None)
MousePointer	0 - Default
MultiLine	0 - False
Name	TextBox1
PasswordChar	
ScrollBars	0 - None
SelectionMargin	-1 - True

Properties - Textbox Control

View Code can be an important control management option because it provides instant access to that portion of the form's script devoted to this control. After you add the control to the form, it becomes a programmable object. Here's an example:



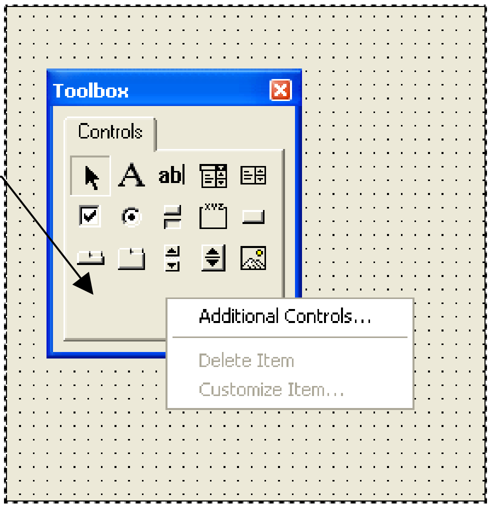
```
Sub TextBox1_AfterUpdate()
    msgbox TextBox1.Value
End Sub
```

How to Add Controls to the Development Toolbox

The default contents of the **Development Toolbox** are intentionally limited to a few controls. However, other controls are available – and you can add them without difficulty:

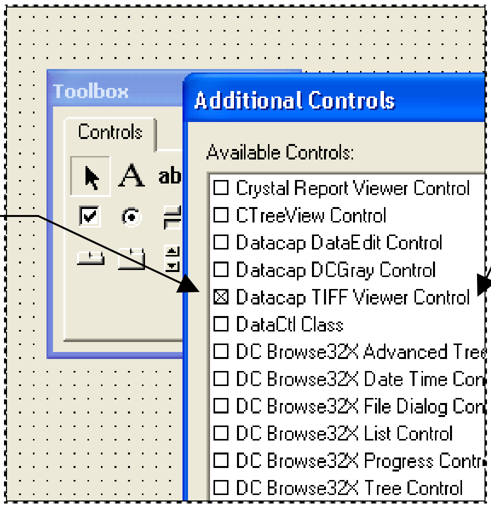
Step	Action
1.	Right-click in the tools section of the toolbox itself.

Right-click in this area.



Step	Action
2.	Select **Additional Controls** to access the ***Additional Controls*** dialog.
3.	Check one or more controls from the **Available Controls** list, and press the OK button

Check



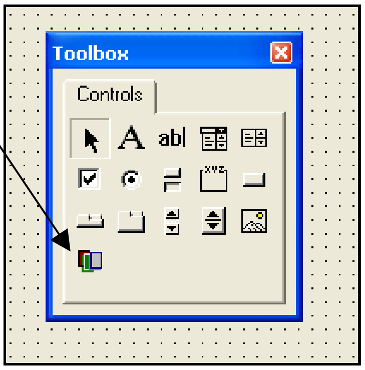
Available Controls:
- ☐ Crystal Report Viewer Control
- ☐ CTreeView Control
- ☐ Datacap DataEdit Control
- ☐ Datacap DCGray Control
- ☒ Datacap TIFF Viewer Control
- ☐ DataCtl Class
- ☐ DC Browse32X Advanced Tre
- ☐ DC Browse32X Date Time Con
- ☐ DC Browse32X File Dialog Con
- ☐ DC Browse32X List Control
- ☐ DC Browse32X Progress Contr
- ☐ DC Browse32X Tree Control

Additional control (?)

To Add a New Control to the Toolbox (continued)

Step	Action
4.	Confirm that an icon representing the control you selected is now in the **Development Toolbox**.

New control



Scripts of the Datacap Form

A Datacap form (.dcf) consists of controls and their properties –and the code that determines the attributes and responses of the form and of the controls.

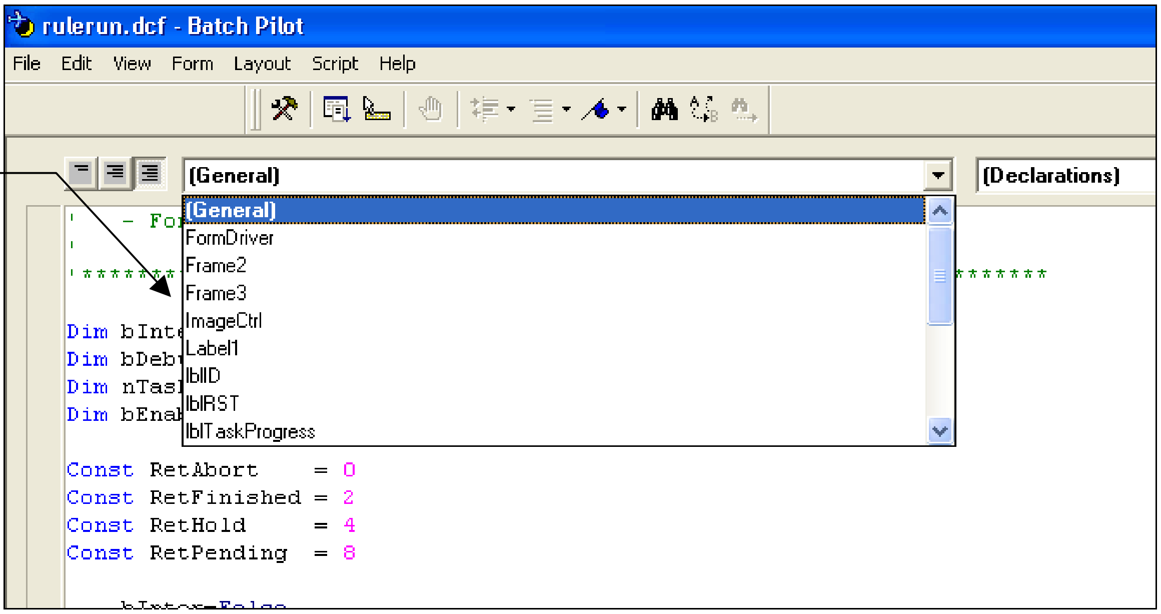
A *Batch Pilot* script is written in *VBScript* with full access to:

- Windows Script Host;
- Built-in and scripting objects of *VBScript;*
- Objects of the Datacap Application Programming Interface (API).

The scripting of a Datacap form takes place at two levels – the overall form and the controls that are part of the form. The script is events-driven.

✓ To access a form's script, open the form within the ***Batch Pilot Window*** and select **View Code** from the **Form** menu. For a look at a specific form's *Batch Pilot* script, open the **rulerun.dcf** form in the **BPilot** folder's **RuleRun** sub-folder.

Controls and Objects



```
'     - Fo
'
'*******                                              *******

Dim bInte
Dim bDeb
Dim nTas
Dim bEna

Const RetAbort    = 0
Const RetFinished = 2
Const RetHold     = 4
Const RetPending  = 8

    bInter=False
```

Batch Pilot Window – *Scripting*

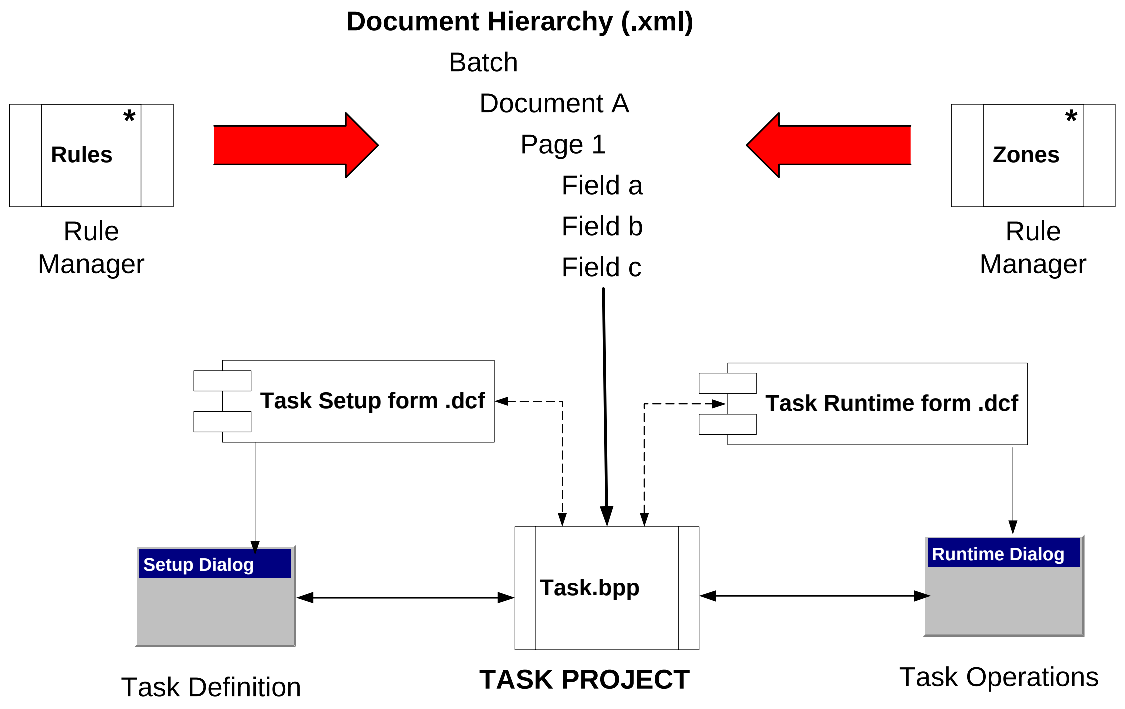
Alert! You cannot copy a control's existing sub-routines into the code of another control. You can, however, copy and paste a portion of a sub-routine of one control into the sub-routine of another control. Although you can copy and paste a complete sub-routine into the `[General]` sector of a form's code, this sector will not retain the code when you save the form.

✓ ***Remember:*** The *Guide to Batch Pilot* describes tools and locations of *Batch Pilot's* scripting process but does ***not*** describe the process itself.

Developing Datacap Forms

The chart below emphasizes three factors to consider in the design and development of a new Datacap form (.dcf):

- Although it is an independent entity, a form only has value when it has been linked to an application's Task Project (.bpp).
- The application's Document Hierarchy is the central property of the Task Project and, by extension, of the forms its employs.
- The Task Definition process uses the ***setup*** form's dialog to acquire operating parameters and settings, and feeds this information to the Task Project. The Task Project, in turn, responds to this data as it manages the task's use of the ***runtime*** form's dialog.



✓ Consider, first, the underlying role of the application's Document Hierarchy. The Document Hierarchy is the backbone of a *Taskmaster* configuration:

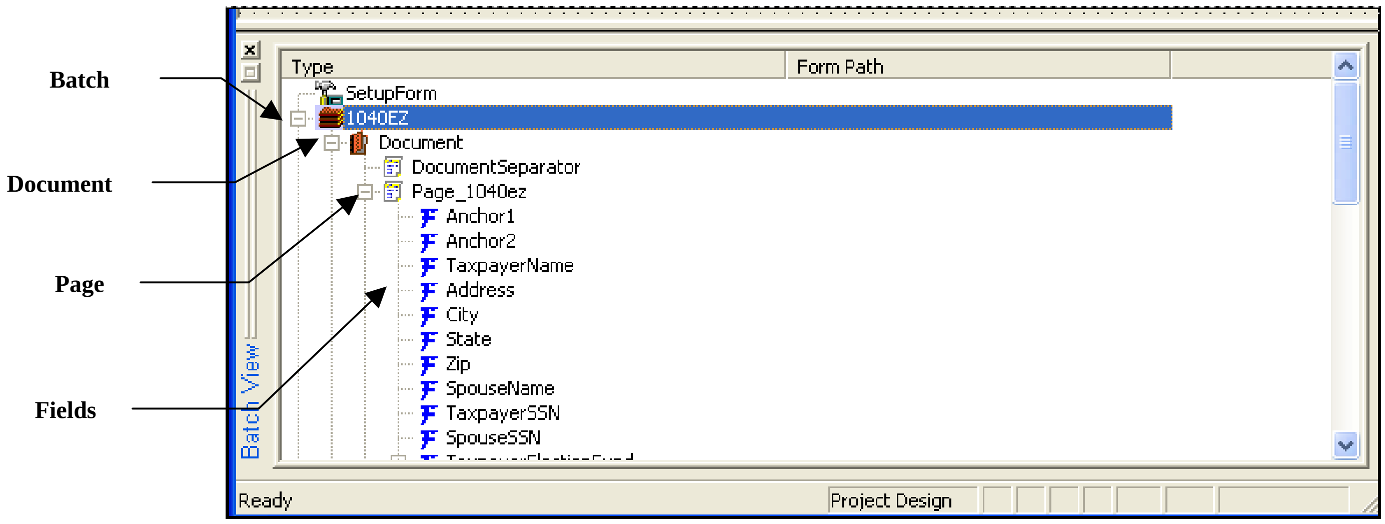
- ♦ Its levels – **Batch**, **Document**, **Page** and **Field -** form a compact, reliable structure that is shared by all elements within the application.
- ♦ **Objects** at a particular level designate processing units for that level.
- ♦ Each object's ***setup*** properties provide it with an identity, and define certain aspects of the object's role.
- ♦ An object's ***runtime*** properties accumulate the data that is processed during task operations.

The Document Hierarchy is also a **“setup DCO**” file: **1040EZ.xml** is in the **Process** directory of the **1040EZ** application folder of your **Datacap** directory.

Chapter 2 of the *Taskmaster Administrator’s Guide* introduces the Document Hierarchy as a cornerstone of the *Taskmaster* architecture; Chapter 3 of the *Guide to Taskmaster Rules* thoroughly explains all facets of the Document Hierarchy.

For **stock** forms, the application’s Document Hierarchy has no relevance until the form becomes part of a Task Project that belongs to the application. (The ***Data Entry*** panel is an exception because its initial structure mirrors the content of the Document Hierarchy – see Chapter 5 for details.)

However, a Task Project ***cannot*** be set up without a Document Hierarchy (Chapter 4). As a result, when you open the project, the hierarchy and its objects are immediately available in the **Setup Tree** of the ***Batch Pilot Window’s*** **Batch View** area. For example, when you open the *1040EZ* RuleRunner Task Project (**rulerun.bpp**) and disconnect (!) its forms, the Document Hierarchy appears with its **Batch**, **Document**, **Page** and **Field** objects.



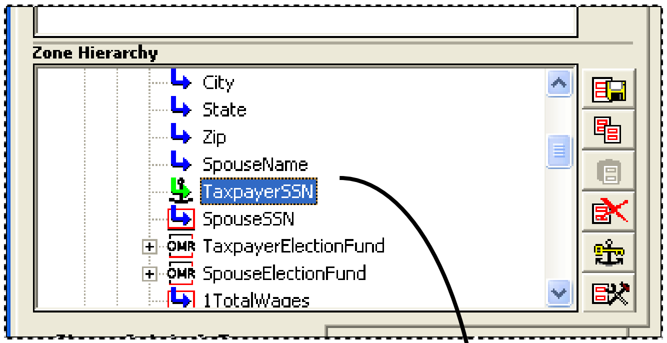
```
SetupForm
1040EZ
  Document
    DocumentSeparator
    Page_1040ez
      Anchor1
      Anchor2
      TaxpayerName
      Address
      City
      State
      Zip
      SpouseName
      TaxpayerSSN
      SpouseSSN
```

1040EZ Document Hierarchy -
Batch View Area of the Batch Pilot Window

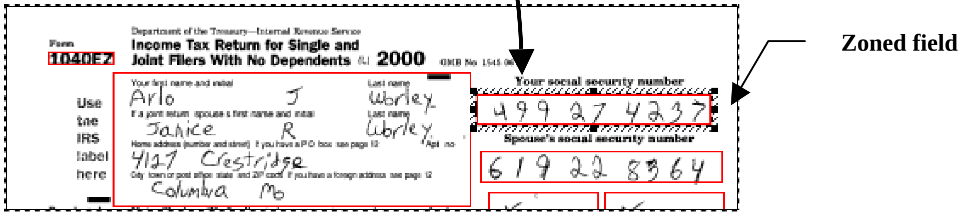
✓ This hierarchy is important because:

- ♦ The application’s *Rule Manager* uses the hierarchy to **zone** fields on a page and assign recognition parameters to the fields (illustrated on the next page);
- ♦ *Rule Manager* also defines the **rules** that the project’s task will apply to objects at each level as it processes a batch, its documents and their pages, and the data in the fields on each page.

Below, zoning the *TaxpayerSSN* field (as an example) determines its location within the *1040EZ* “fingerprint”:

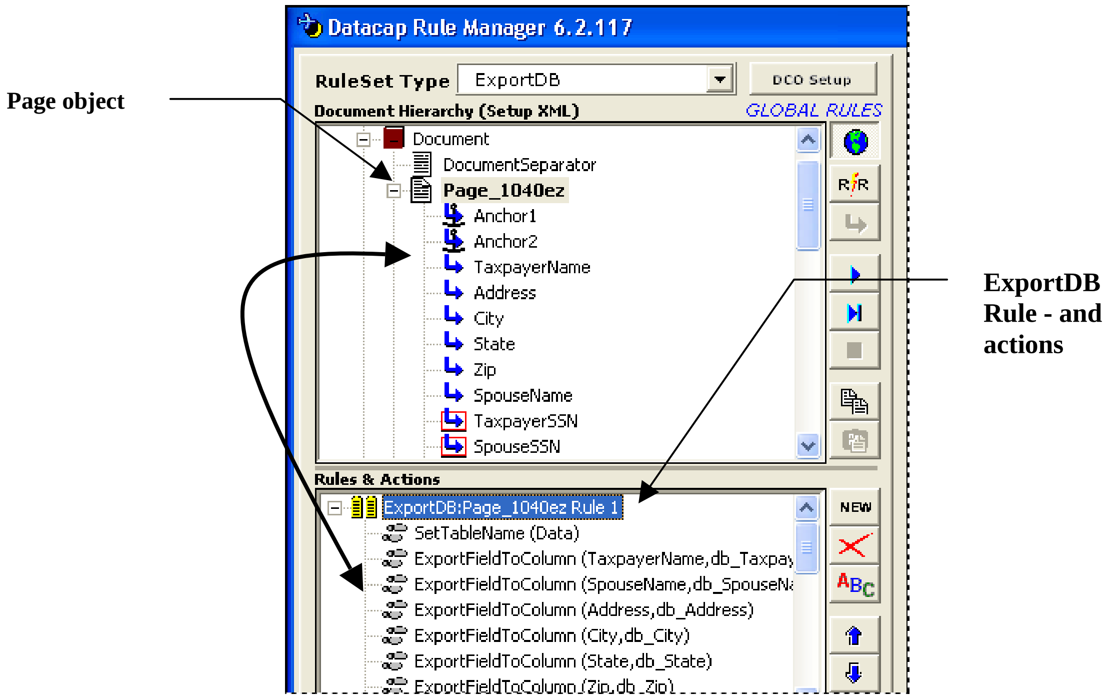


Rule Manager – *Zone Hierarchy*


Zoned field
Form 1040EZ
Department of the Treasury—Internal Revenue Service
Income Tax Return for Single and Joint Filers With No Dependents
2000
Your social security number
499 27 4237
Spouse's social security number
619 22 8364
Arlo J Ubrley
Janice R Ubrley
4127 Crestridge
Columbia Mo

Zoned TaxpayerSSN Field

Here, an **ExportDB** rule has been assigned to the *Page_1040EZ* **Page** object: as a result, the rule will export values in each field specified by the set of **ExportFieldToColumn** actions that make up the rule.



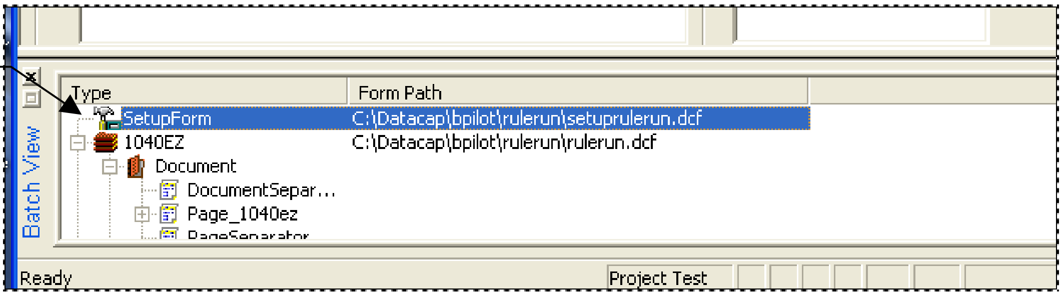
ExportDB:Page_1040ez Rule 1
SetTableName (Data)
ExportFieldToColumn (TaxpayerName,db_Taxpay
ExportFieldToColumn (SpouseName,db_SpouseNa
ExportFieldToColumn (Address,db_Address)
ExportFieldToColumn (City,db_City)
ExportFieldToColumn (State,db_State)
ExportFieldToColumn (Zip,db_Zip)

ExportDB Rule

The *1040EZ* Export task is a *RuleRunner* task: it applies the pre-defined rules that determine which fields will supply the data in a batch to an Export database.

The ***setup*** form of the Export Task Project (**RRExport.bpp**) is a **stock** form (**setuprulerun.dcf**):

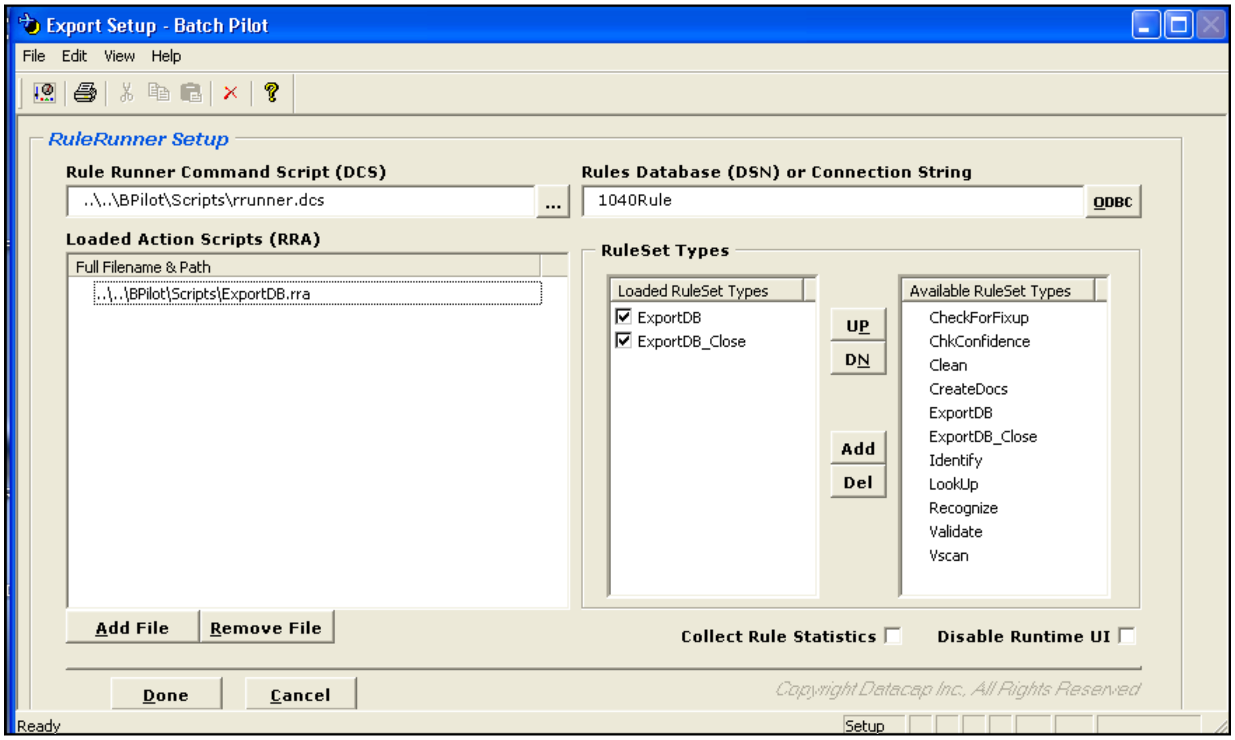
Setup form



Type	Form Path
SetupForm	C:\Datacap\bpilot\rulerun\setuprulerun.dcf
1040EZ	C:\Datacap\bpilot\rulerun\rulerun.dcf
Document	
DocumentSepar…	
Page_1040ez	
PageSeparator	

RRExport Task Project

During Export Task Setup, this form becomes the ***Task Setup*** dialog that accumulates the information that the Task Project will pass on to the task as it processes a batch:



Rule Runner Command Script (DCS): ..\..\BPilot\Scripts\rrunner.dcs

Rules Database (DSN) or Connection String: 1040Rule

Loaded Action Scripts (RRA)

Full Filename & Path: ..\..\BPilot\Scripts\ExportDB.rra

RuleSet Types

Loaded RuleSet Types:
- ☑ ExportDB
- ☑ ExportDB_Close

Available RuleSet Types:
- CheckForFixup
- ChkConfidence
- Clean
- CreateDocs
- ExportDB
- ExportDB_Close
- Identify
- LookUp
- Recognize
- Validate
- Vscan

Collect Rule Statistics ☐ Disable Runtime UI ☐

1040EZ - Export Task Setup dialog

The settings in two fields are especially important:

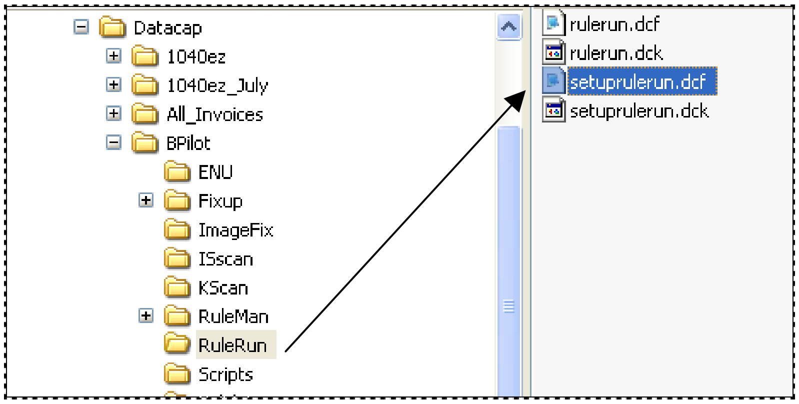
- **Loaded RuleSet Types** lists the RuleSet Types that include the rules that the task is to carry out.
- **Rules Database** identifies the database that contains a full set of information about the *1040EZ* application's *Rule Manager* and rules.

Setup Forms and Dialogs

The **setuprulerun.dcf** form is a **stock** form: the form is not affiliated with a specific application ***until*** you link it to a Task Project that belongs to an application.

As soon as you assign the form to a Task Project, it is available for the setup of a task based on the Task Project.

For a closer look at the **setuprulerun.dcf** form, access *Batch Pilot* and open the form: it is in the **RuleRun** sub-folder of the **Datacap** directory's **BPilot** folder. (The accompanying **.dck** file holds the form's **Hot Key** specifications. For details, see Page 39).



The **stock** *RuleRunner* ***setup*** form, as an example, consists of:

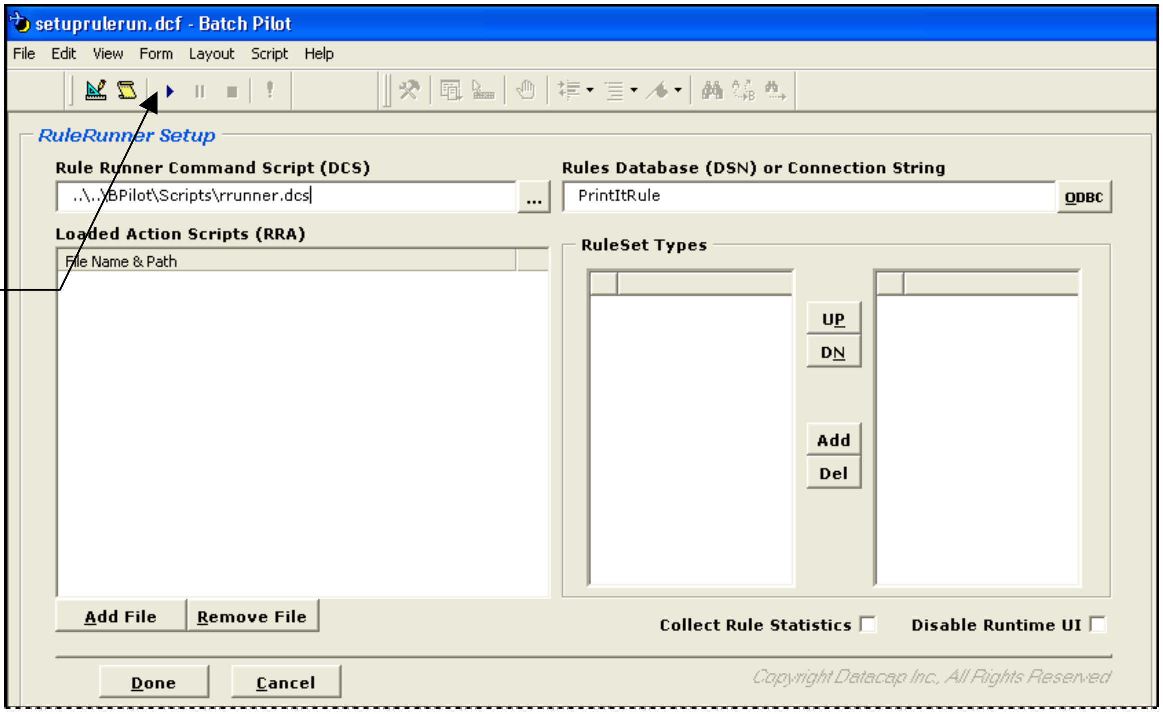
- Controls that include the **User Form** container
- Properties assigned to each control
- VBScript code that determines the role of each control when the form is part of a Task Project and, eventually, serves a ***Task Setup*** dialog.

✓ ***Don't hesitate*** to practice with any *1040EZ* form, its controls and its code, as long as you are working in a Test environment. *Batch Pilot* makes it easy for you to modify these elements and evaluate your changes:

- Open the form in the ***Batch Pilot Window***.
- While the form is in *Design* mode (select **Design** from the **Form** menu) highlight a control.
- Add or alter a control, or change a control's properties or code (Page 19).
- Click on the **Run Script** icon of the **Form** toolbar (Chapter 2) to assess the impact of your changes (illustrated on the next page.)

Important! After you click on the **Run Script** icon, you can try out many of the existing controls – even those you haven't touched.

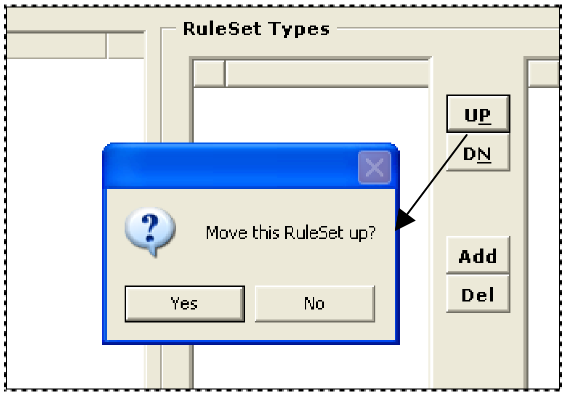
Click here to test.


setuprulerun.dcf - Batch Pilot
File Edit View Form Layout Script Help
RuleRunner Setup
Rule Runner Command Script (DCS)
..\..\BPilot\Scripts\rrunner.dcs
Rules Database (DSN) or Connection String
PrintItRule
ODBC
Loaded Action Scripts (RRA)
File Name & Path
RuleSet Types
UP
DN
Add
Del
Add File
Remove File
Collect Rule Statistics
Disable Runtime UI
Done
Cancel
Copyright Datacap Inc., All Rights Reserved

RuleRunner Setup Form

Suppose you decide to add a warning to the functionality of the Up button in the **RuleSet Type** area.

Inserting this change involves a minor change to the Up button control's script. Afterwards, if you press the ▶ tool, and ***then*** the UP button, the scripted warning - or a debugging error message - will appear:



Success!

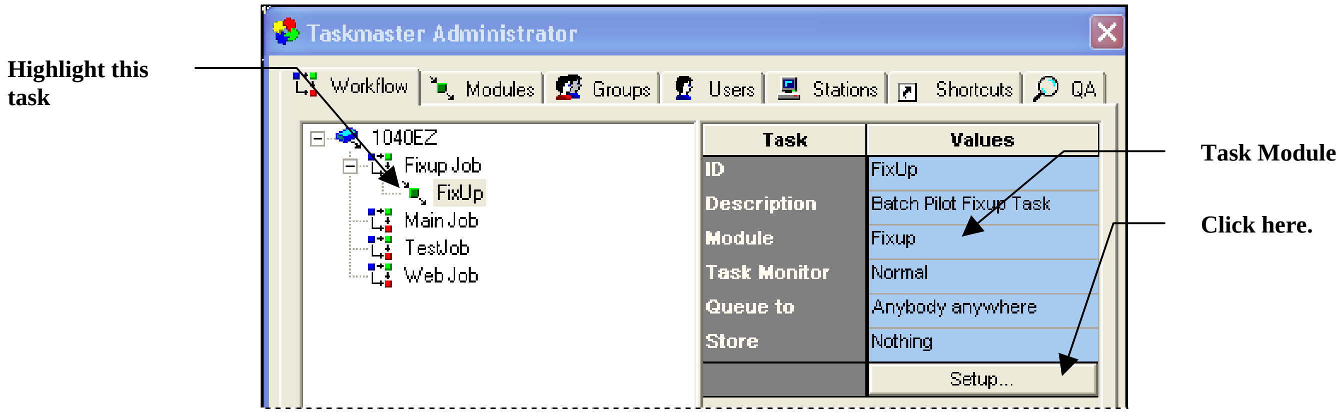
Controls of a Setup Form

The controls of a ***setup*** form combine in a dialog that solicits an individual task's setup information.

The compact *RuleRunner* ***setup*** form on the previous page is a good example of how buttons work closely with data fields:

- The ODBC button accesses the Windows ***ODBC Data Source Administrator*** and a list of **D**ata **S**ource **N**ames (DSN) that includes the DSN for the current application's Rules database.
- During Task Setup, when the **Rules** database is identified, the right-hand field in the **RuleSet Types** frame will be filled with the names of the **Available RuleSet Types** that include rules that have been defined for the current application.
- The Add button control moves one or more RuleSet Types into the **Loaded RuleSet Types** field on the left. As a result, the task will apply previously defined rules of these types when it processes a batch.
- On the left, file management button controls will locate and install Actions files (.rra) and a generic *RuleRunner* processing script (.dcs). A *RuleRunner* task needs both to carry out its work.
- Two checkbox controls represent processing options.
- The Done and Cancel buttons will conclude this part of the Task Setup process.

The setup of other types of tasks requires more extensive information and, therefore, a more comprehensive ***setup*** form and more sophisticated controls. The FixUp **task** of the *1040EX* FixUp **job** is a good example:



Task	Values
ID	FixUp
Description	Batch Pilot Fixup Task
Module	Fixup
Task Monitor	Normal
Queue to	Anybody anywhere
Store	Nothing

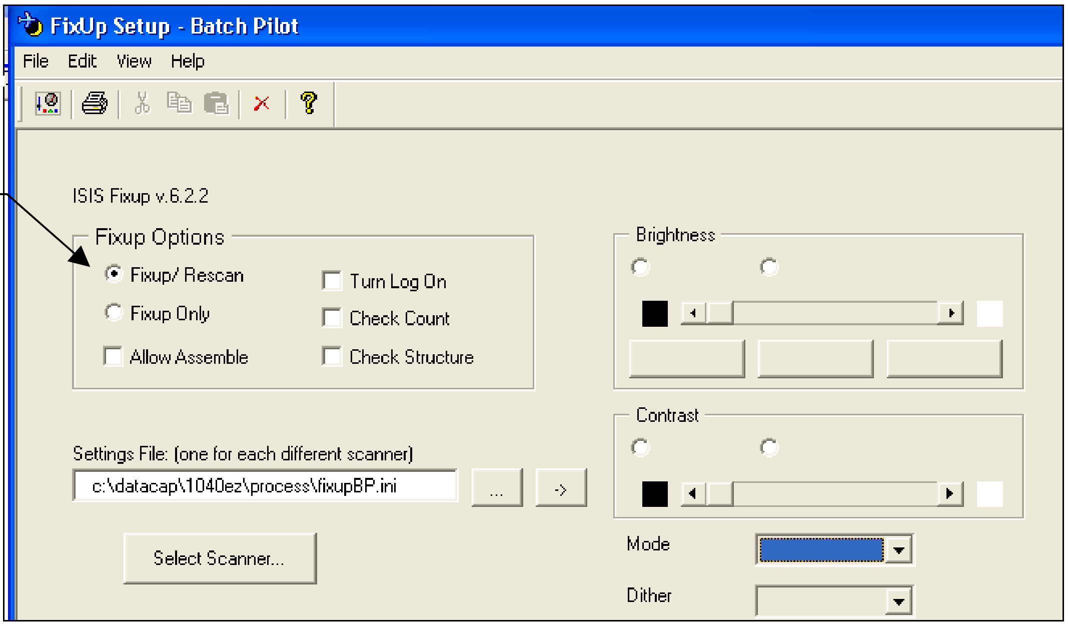
1040EZ Taskmaster Administrator – *Workflow tab*

(Unless you are physically linked to a scanner at this point and have assembled a Source Device for the scanner, the setup process will note this apparent deficiency - but you can probably proceed anyway.)

By default, the *1040EZ* FixUp task employs an ISIS driver. (Chapter 11 of the *Guide to Taskmaster Rules* covers the setup and operation of FixUp tasks using ISIS and Kofax drivers.)

✓ Select the **FixUp/Rescan** option below to ensure that controls associated with scanner selection and criteria appear in the dialog. Otherwise, the scope of the ***Task Setup*** dialog is more limited.

Select this option



ISIS Fixup v.6.2.2

Fixup Options
- [✓ Fixup/ Rescan]
- Fixup Only
- ☐ Allow Assemble
- ☐ Turn Log On
- ☐ Check Count
- ☐ Check Structure

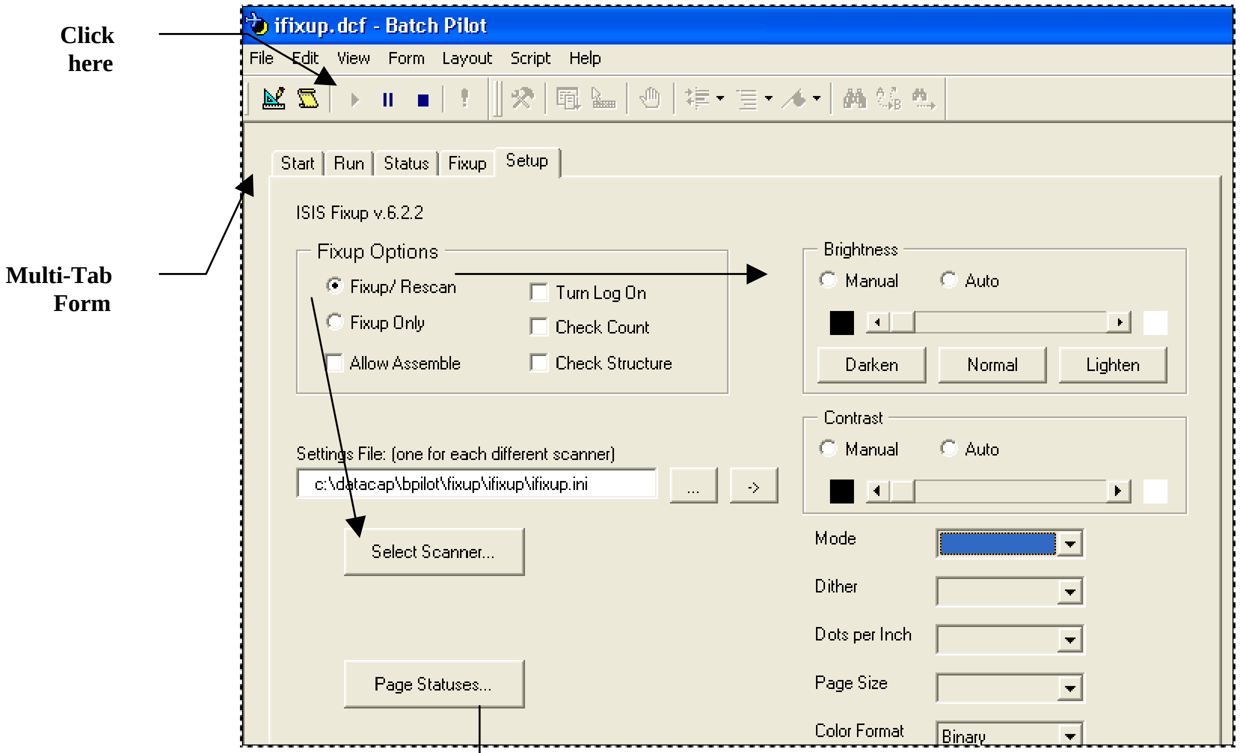
Settings File: (one for each different scanner)
c:\datacap\1040ez\process\fixupBP.ini

1040EZ FixUp Task – Setup Parameters (ISIS Driver)

☛ Be sure to study this dialog in combination with the Datacap form that is responsible for it: **ifixup.dcf** is in the **iFixUp** sub-folder of the **BPilot** folder's **FixUp** directory. Use the tools of the ***Batch Pilot Window*** to open the form (illustrated on the next page.)

- The iFixUp form is a single Multi-tab Form (Page 33) – with tabs that provide ***setup*** parameters (*Setup* and *Status*) as well as ***runtime*** information and mechanics (*Start*, *Run* and *FixUp*.)
- You can experiment with the form and its controls by pressing the window's ▶ icon, and the controls in each tab.

A number of controls in this ***setup*** form have an immediate impact on the form when it becomes a Fixup task's dialog.

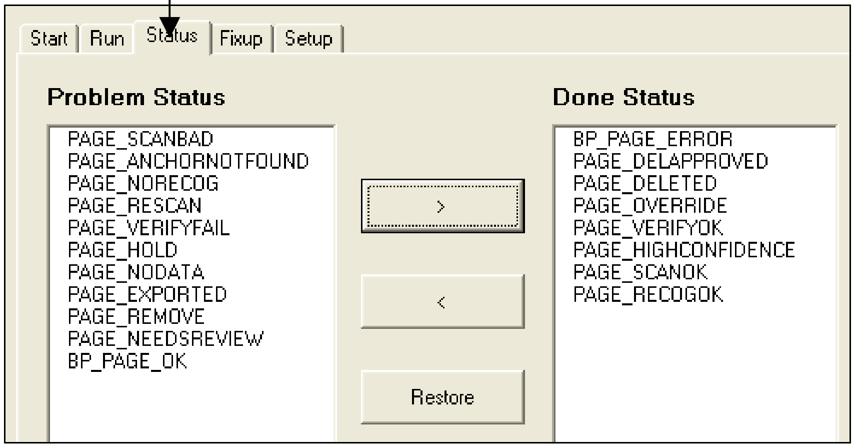


ISIS Fixup v.6.2.2

Settings File: (one for each different scanner) c:\datacap\bpilot\fixup\ifixup\ifixup.ini

Color Format: Binary

ISIS FixUp Multi-tab Form



Problem Status

PAGE_SCANBAD
PAGE_ANCHORNOTFOUND
PAGE_NORECOG
PAGE_RESCAN
PAGE_VERIFYFAIL
PAGE_HOLD
PAGE_NODATA
PAGE_EXPORTED
PAGE_REMOVE
PAGE_NEEDSREVIEW
BP_PAGE_OK

Done Status

BP_PAGE_ERROR
PAGE_DELAPPROVED
PAGE_DELETED
PAGE_OVERRIDE
PAGE_VERIFYOK
PAGE_HIGHCONFIDENCE
PAGE_SCANOK
PAGE_RECOGOK

Setup Statuses dialog

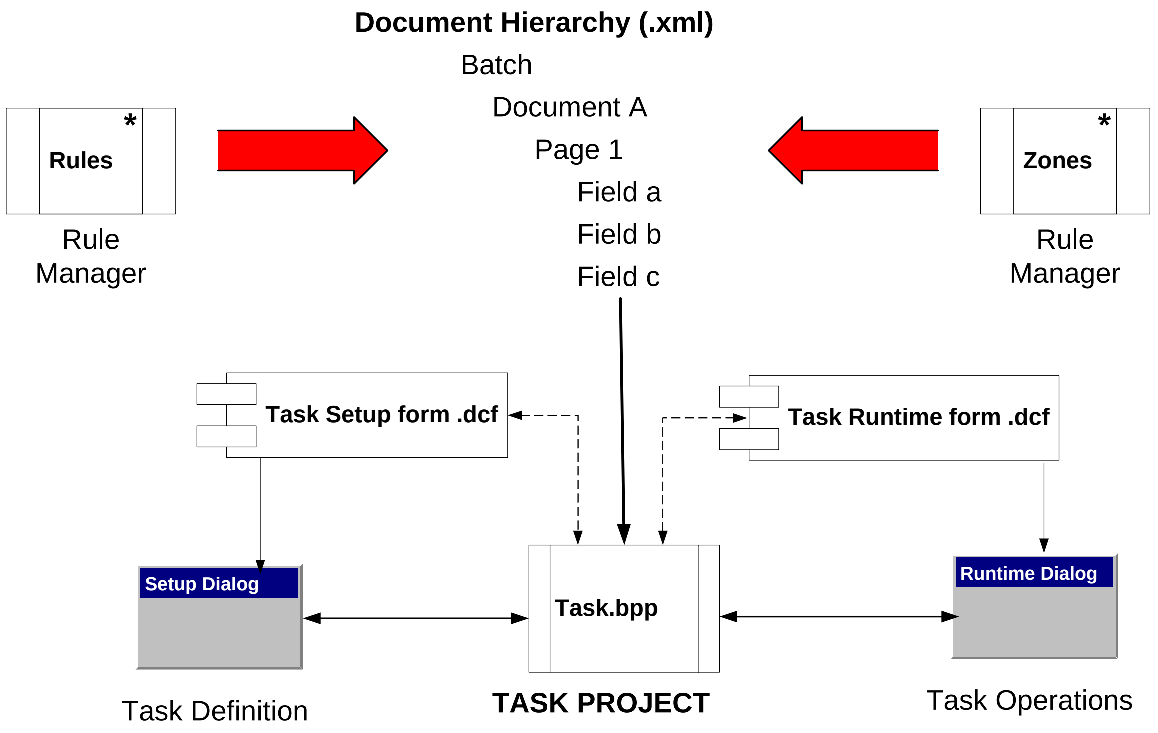
Consider, first, the **FixUp/Rescan** and **FixUp Only** radio buttons in the **FixUp Options** frame. A review of their properties indicates that both are members of the **FMode** (*FixUpMode*) Group of radio button controls. An equally brief investigation of the code behind these controls reveals that:

- The Click event of *opFixUpRescan* contol (**FixUp/Rescan** radio button) activates the various scanner controls on the right; as well as the Select Scanner button.
- The Click event of *opFixUp* control (**FixUp Only** radio button) hides the scanner controls on the right, as well as the Select Scanner button.

- When the form is in *Test Mode* – after you have pressed the ▶ icon – the Select Scanner button attempts to locate a scanner that has been connected to your computer – and will proceed with a test even if it cannot find the scanner.
- If you select the **FixUp/Rescan** radio button, you can investigate the properties of the various scanner controls on the right – and the code behind each control.
- The Page Statuses button opens the S***etup Statuses*** dialog that assigns available Page Statuses to two categories (illustrated on the previous page). ***Alert!*** This dialog is also the *Statuses* tab of the Multi-tab Form: Page 33 explains how this form's tabs become contributing dialogs

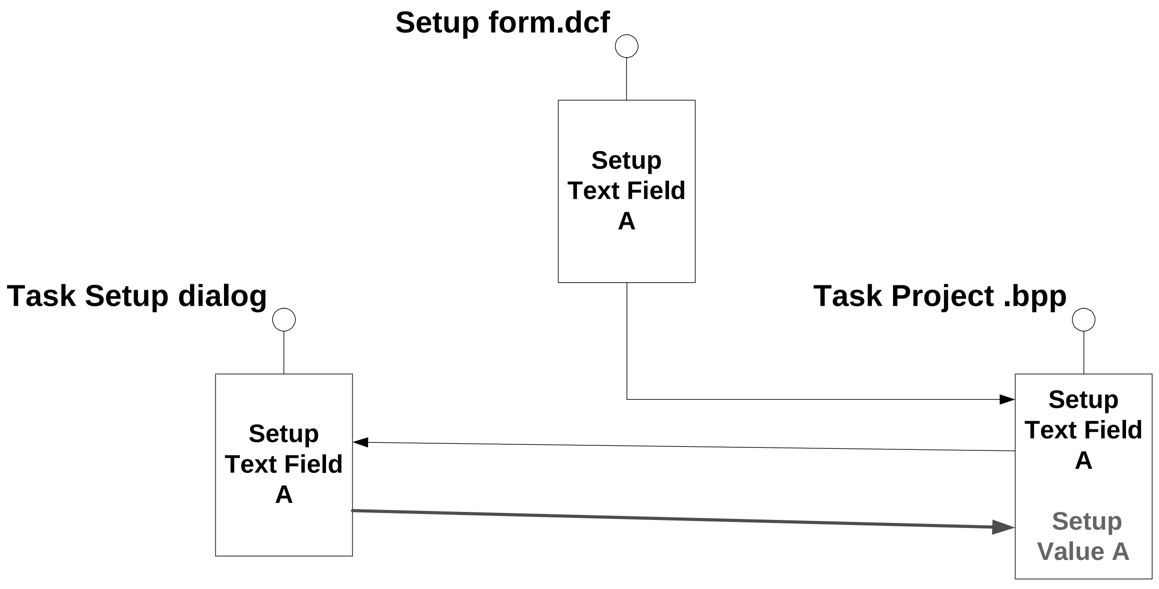
How to Construct a Setup Form

The lower left-hand side of this diagram stresses the relationship between a ***setup*** form, a Task Project that belongs to an application, and a Task Definition that accesses the setup form through the Task Project - as the ***Task Setup*** dialog:



✓ You may well conclude that your application does not need any new ***setup*** forms…that the **stock** forms in the **Datacap** directory's **BPilot** folder will meet the setup needs of your Task Projects and Task Definitions (Chapter 4).

If, however, you do decide to put together a new form or modify an existing form, the form and its controls must meet the basic requirements outlined in the chart on the next page.



This basic example depicts a ***setup*** form (.dcf) with one control: *Setup Text Field A.*

When the ***setup*** form is attached to a Task Project (.bpp), *Setup Text Field A* instantly becomes a ***setup*** property of the Task Project (Chapter 4).

The Task Definition of a task based on this Task Project places a value in *Setup Text Field A* when it appears in the ***Task Setup*** dialog.

This step assigns the value to the Task Project (.bpp), which can then make it available to the task's ***runtime*** operations.

✓ ***Very Important!*** As Chapter 4 explains, the Task Project is a Settings file (.bpp) that contains ***setup*** and ***runtime*** criteria to govern a task's configuration steps and operations. In this example, the value in *Text Field A* is a setup parameter. This means that the field itself will appear in the ***Task Setup*** dialog as soon as the form has been attached to the appropriate Task Project. However, a value that an Administrator enters in *Text Field A* cannot be picked up by the Task Project (.bpp) unless the ***runtime*** form's code for the **Text Field A** field control includes a function that employs the **SetProfileString** method of *Taskmaster's* **Pilot** object. Here's an example:

```
Sub SaveSetupData()
Call Pilot.SetProfileString(“SetupA”,”FieldA”,TextA.Value,
     Pilot.ProjectPath)
End Sub
```

This example of the **Pilot_FormSetup** function is intentionally limited; a typical ***setup*** form will use the function to assign many more values to the settings of a Task Project (.bpp).

In the example:

- ♦ "SetupA" identifies a `[sector]` of the Task Project's settings – or adds the sector if necessary;
- ♦ "FieldA" specifies a setting within the `[SetupA]` sector…or adds the setting.
- ♦ `TextA.Value` specifies the source of this setting's value (or is a default value.)
- ♦ `Pilot.ProjectPath` directs the function to add new or modified values to the Task Project to which the setup form is attached.

✓ ***Alert!*** This brief example will not work unless you provide the form's code with an event of some sort that actually calls the `SaveSetupData()` function on the previous page. Something as simple as the Click event of a ***setup*** form's Done button:

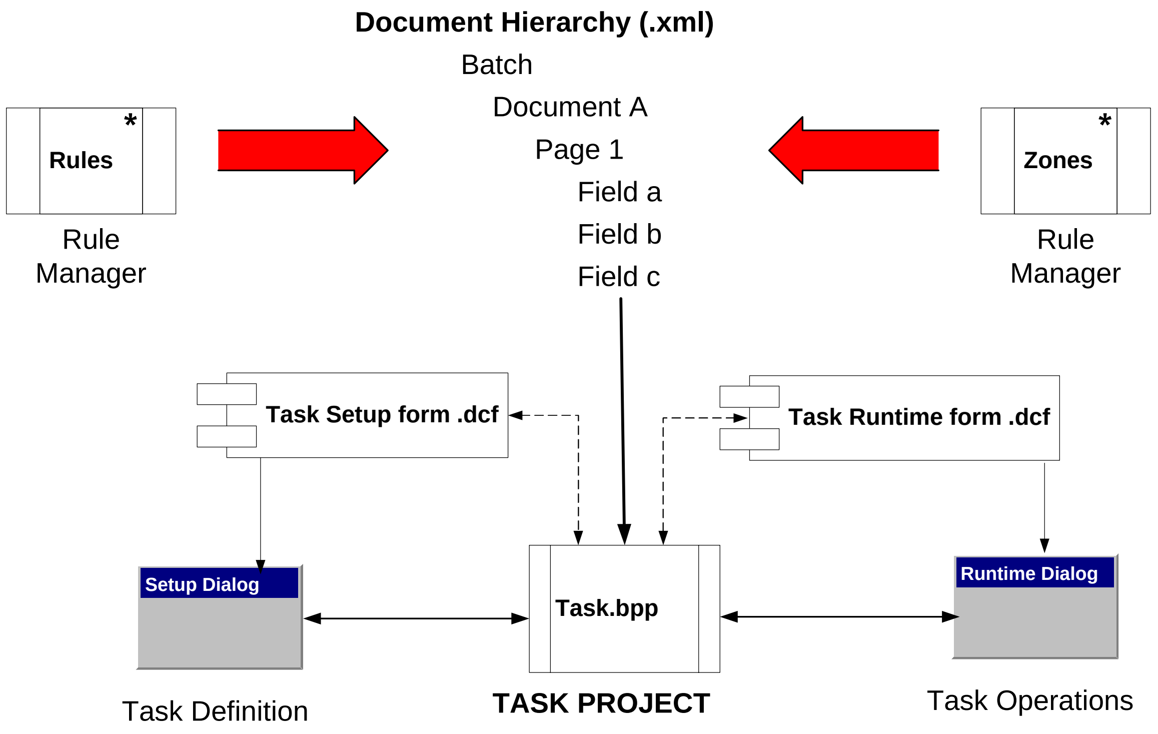
```
Sub btnDone_Click()
   Call SaveSetupData()
   Pilot.SetupStop(1)
End Sub
```

✓ If you are ready to explore the basics of the code that underlies a ***setup*** form, open the **setuprulerun.dcf** form and use the tools of the ***Batch Pilot Window*** to review this valuable form's code.

Runtime Forms and Dialogs

A task's ***runtime*** dialog will appear on an operator's screen as soon as the operator launches the task if:

- Its Datacap form (.dcf) has been bound to the **Batch** object of the application's Document Hierarchy – or, in the case of a ***Data Entry*** panel – to a **Page** object.
- A Task Definition using a dialog based on the Task Project's ***setup*** form is complete.

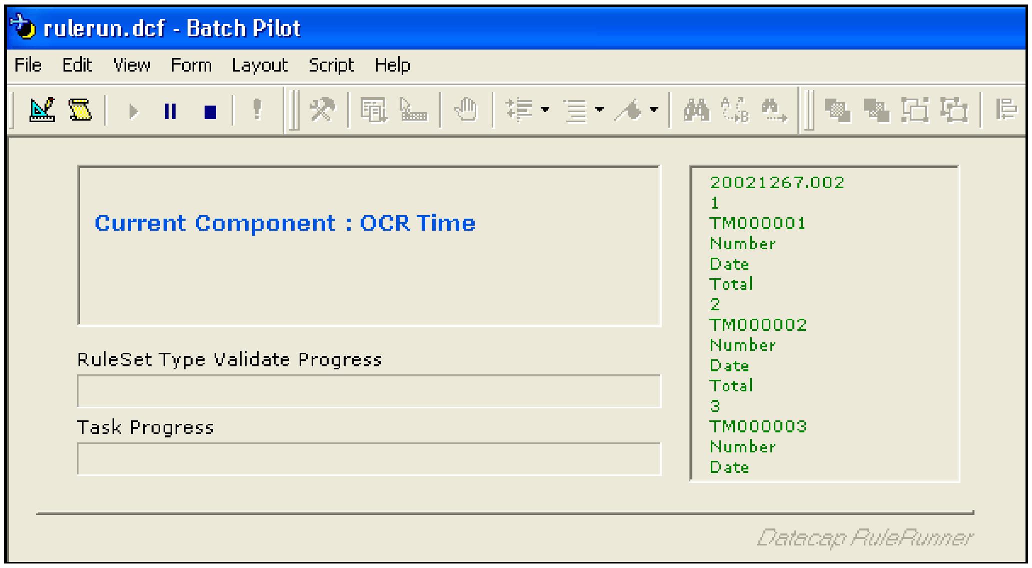


In response to current specifications in the Task Project, a ***runtime*** form's dialog solicits and displays information, and updates the files that contain batch information.

The **stock *runtime*** form of a *RuleRunner* task – **rulerun.dcf -** is relatively passive and even hidden (if the ***setup*** form's **Disable Runtime UI** option has been checked – Page 17.)

For a comprehensive look at this ***runtime*** form and its controls, use the **Open Form** item of the ***Batch Pilot Window's*** **File** menu to open it directly (it's in the **BPilot** folder's **RuleRun** sub-folder).

✓ Be sure to go back and forth between *Design* mode and *Form Test* mode to identify and investigate the form's controls – and its runtime appearance.


rulerun.dcf - Batch Pilot
File Edit View Form Layout Script Help
Datacap RuleRunner

Current Component : OCR Time

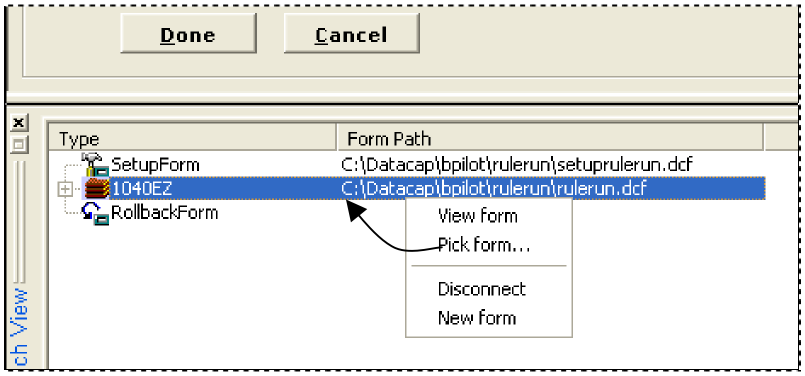
RuleSet Type Validate Progress

Task Progress

```
20021267.002
1
TM000001
Number
Date
Total
2
TM000002
Number
Date
Total
3
TM000003
Number
Date
```

RuleRunner Runtime Form (rulerun.dcf)

As part of the definition of the *1040EZ* application's Export Task Project (**Export.bpp)**, this form was linked directly to **Batch** object of the *1040EZ* Document Hierarchy:



Type	Form Path
SetupForm	C:\Datacap\bpilot\rulerun\setuprulerun.dcf
1040EZ	C:\Datacap\bpilot\rulerun\rulerun.dcf
RollbackForm	

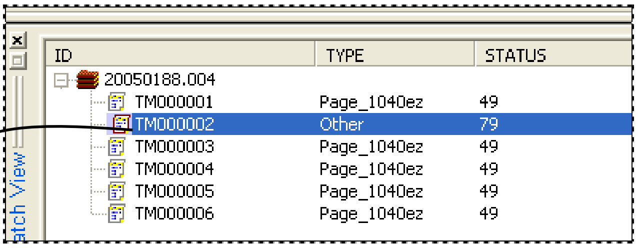
Batch Pilot Window – *Batch View Area*

When the Export task runs, the ***Export*** dialog will automatically display data in each field until the task finishes; the task's operator does not need to participate.

In contrast, the ***runtime*** dialog of the *1040EZ* iFixUp task requires extensive operator participation (see the illustrations on the next page.)

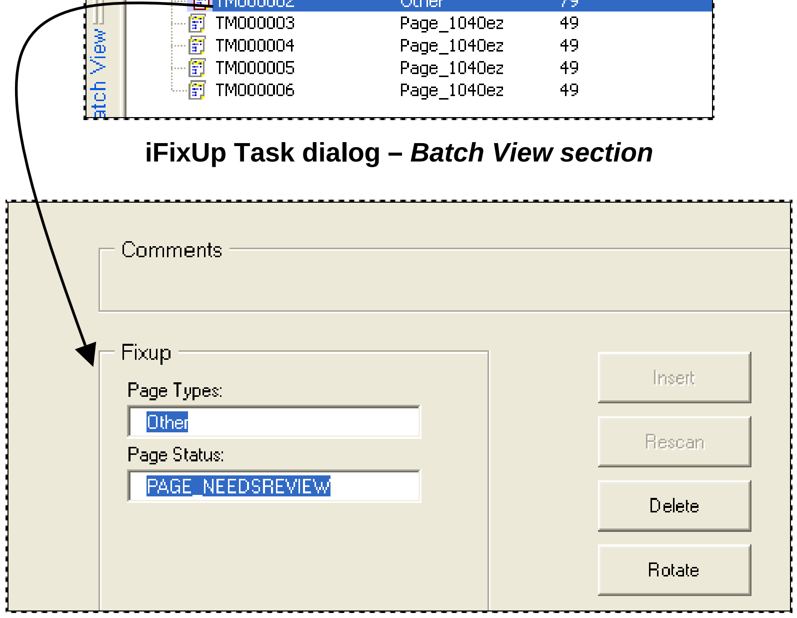
☛ Although the Export task applies a small set of pre-defined rules and runs in the processing background, and the FixUp task uses extensive ***setup*** specifications (Page 33) and an interactive ***runtime*** dialog, both tasks employ **stock** forms (.dcf). These forms are in the applicable sub-folder of the **Datacap** directory's **BPilot** folder. Note, however, that the *FixUp* tab of the FixUp Multi-tab Form (illustrated on the next page) is the foundation for the task's *runtime* dialog. (For more about Multi-tab Forms, see Page 33).

The operator of a Verification task works with a ***Data Entry*** panel produced exclusively for a single application, instead of with a **stock** form. Page 7 introduces the ***Data Entry*** panel; Chapter 5 explores panel development, modification and use.



ID	TYPE	STATUS
20050188.004		
TM000001	Page_1040ez	49
TM000002	Other	79
TM000003	Page_1040ez	49
TM000004	Page_1040ez	49
TM000005	Page_1040ez	49
TM000006	Page_1040ez	49

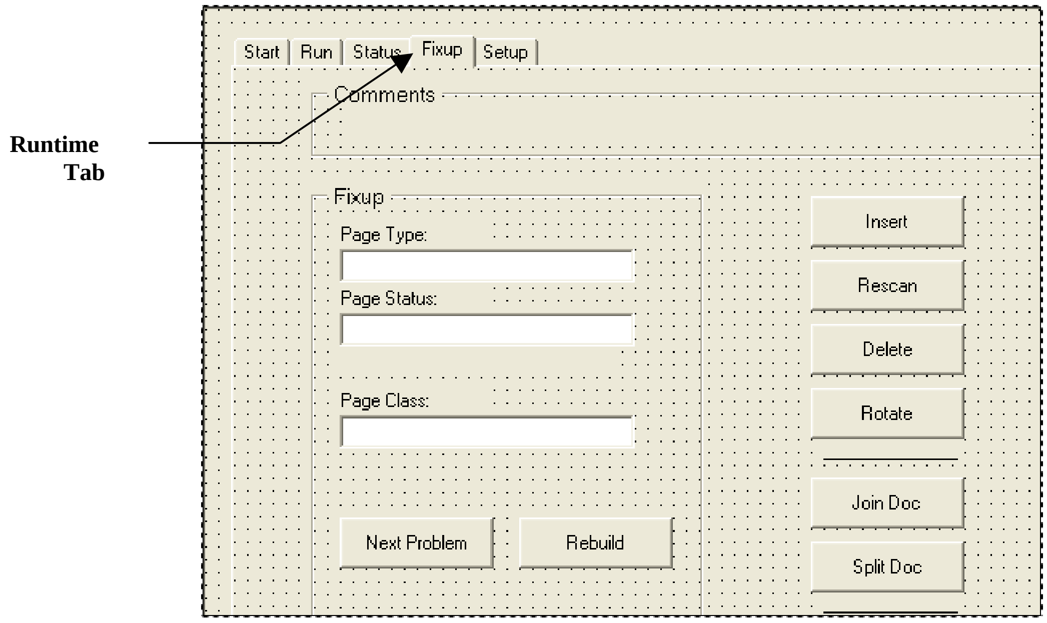
iFixUp Task dialog – *Batch View section*



Page Types: Other

Page Status: PAGE_NEEDSREVIEW

iFixUp Task dialog – *Upper Action Fields and Buttons*



iFixUp Multi-tab Form – *FixUp tab*

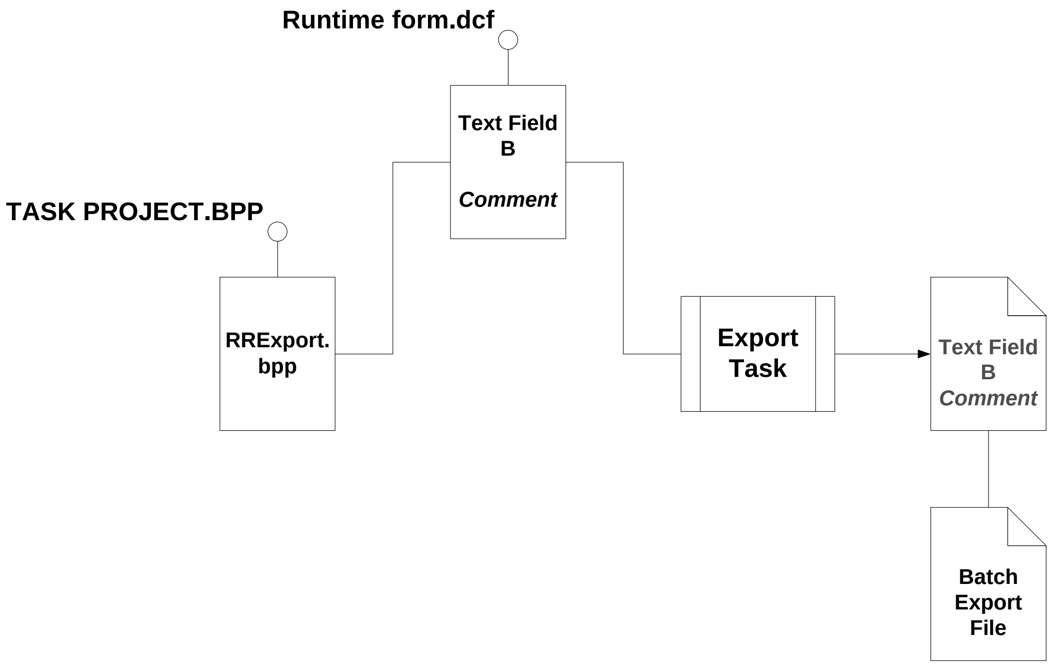
How to Construct a Runtime Form

A ***runtime*** form is the other principal component of a Task Project (Chapter 4). When the task that employs the form's dialog runs, the task solicits information from a number of sources, processes much of this data, and uses the data to update files, databases, and the contents of the current batch.

In the example outlined in the chart below, the developer has:

- ♦ Saved the **stock** *RuleRunner* ***runtime*** form as a new form.
- ♦ Added a **Comment** field control **-** *Text Field B* – to the new form.
- ♦ Assigned the new ***runtime*** form to the Export Task Project (.bpp).

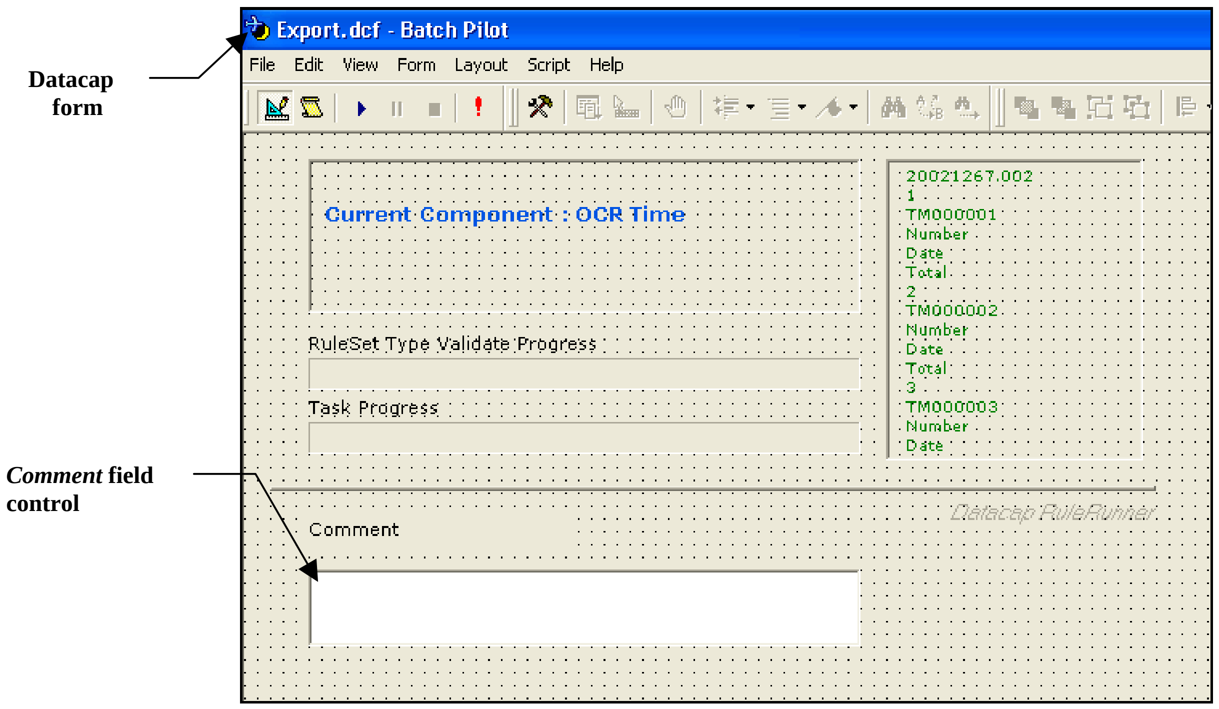
When the Export task runs, the task's operator can enter a brief note in the field. The note's value will be appended to the Export file for the current batch, or entered in the Export database.



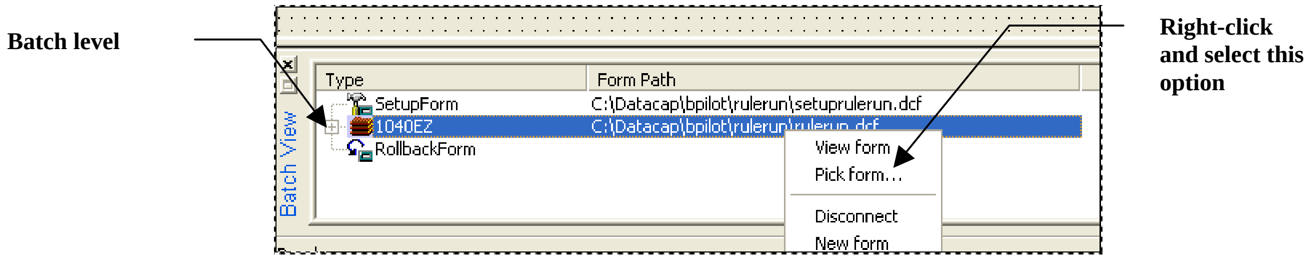
✓ ***Very Important!*** Even this simple modification will ***not*** work unless you:

- Add a **Field** object for the new *Comment* field to the application's Document Hierarchy;
- Add an **Export** action to a new or existing Export rule – an action that will export values in the *Comment* field to the Export file for the current batch – or to an Export database.

The following page briefly reviews these procedures.



New Runtime Form – *Export.dcf*

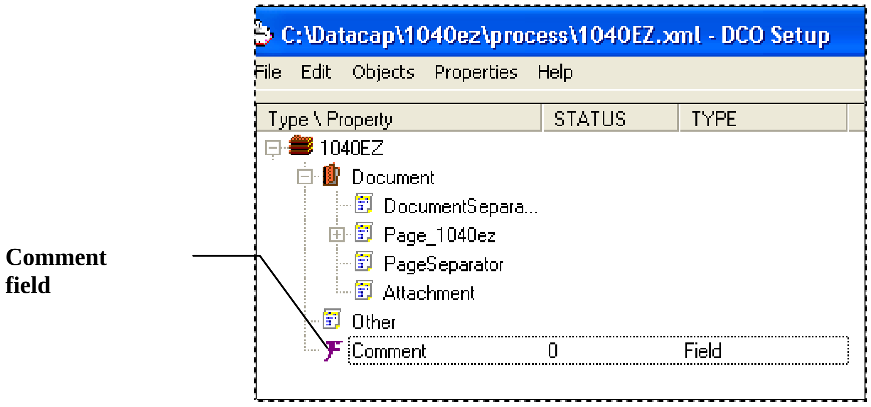

Batch level
Right-click and select this option
Batch View
Type
Form Path
SetupForm
C:\Datacap\bpilot\rulerun\setuprulerun.dcf
1040EZ
C:\Datacap\bpilot\rulerun\rulerun.dcf
RollbackForm
View form
Pick form...
Disconnect
New form

Runtime Form Assignment
<u>Export.bpp Task Project</u>

Batch Pilot helps you create a new ***runtime*** form or modify an existing form without difficulty. (***Remember:*** You can practice with the forms of the *1040EZ* application.)

✓ Chapter 3 of the *Guide to Taskmaster Rules* describes all aspects of a Document Hierarchy, and shows you how to add objects at each level: **Batch**, **Document**, **Page** and **Field**. In this case, the Export task cannot work with the revised form simply because it cannot yet find a *Comment* **Field** object in the Task Project's Document Hierarchy (Chapter 4). And even if it could find the field, it wouldn't know what to do with it!

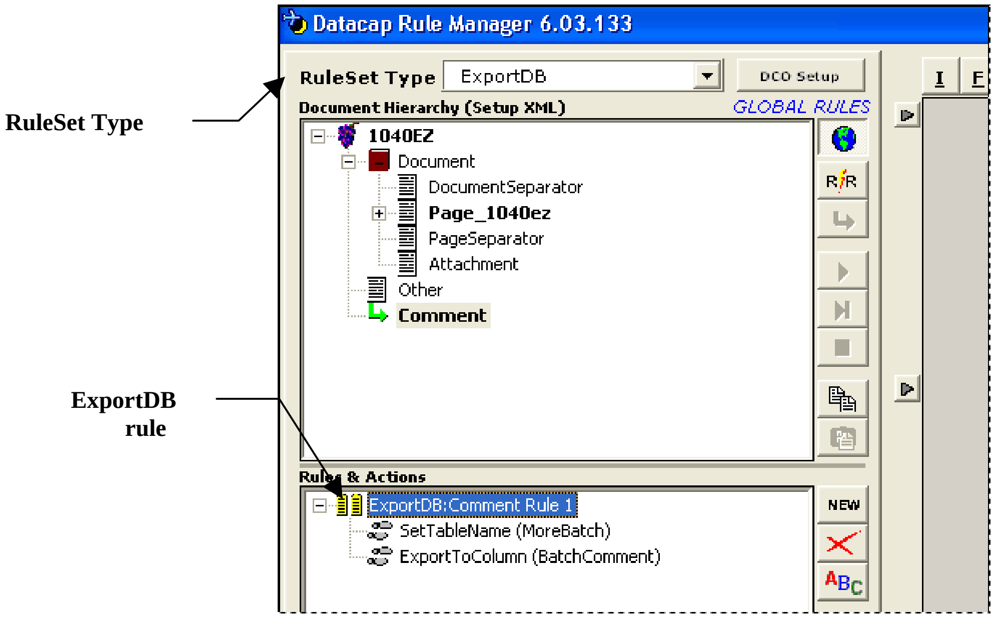
In the illustration on the next page, the *Comment* **Field** object is now a child of the *1040EZ* **Batch** object. This means that the field's value will be appended to batch information in the Export database – rather than to document or page data.


Comment
field
C:\Datacap\1040ez\process\1040EZ.xml - DCO Setup
File Edit Objects Properties Help

Type \ Property	STATUS	TYPE
1040EZ		
Document		
DocumentSepara...		
Page_1040ez		
PageSeparator		
Attachment		
Other		
Comment	0	Field

1040EZ Document Hierarchy

Using the *Rules* panel of the *1040EZ* ***Rule Manager Window***, you can make sure that any value in the field will be exported to the application's Export **database** (in this case):



RuleSet Type: ExportDB

Document Hierarchy (Setup XML)

- 1040EZ
  - Document
    - DocumentSeparator
    - Page_1040ez
    - PageSeparator
    - Attachment
  - Other
  - Comment

Rules & Actions

- ExportDB:Comment Rule 1
  - SetTableName (MoreBatch)
  - ExportToColumn (BatchComment)

ExportDB Rule – *Comment field*

✓ In the example, the **ExportDB** rule will only work if an earlier rule applied to the *1040* **Batch** object has opened the Export database…and the database includes a **MoreBatch** table with a *BatchComment* column.

Two unusual challenges remain:

1. The form's script has to continue to display the form during processing until the operator has a chance to enter a value in the **Comment** field – or take an alternative, acceptable action.

   To meet this requirement, the script will need a combination of statements similar to:

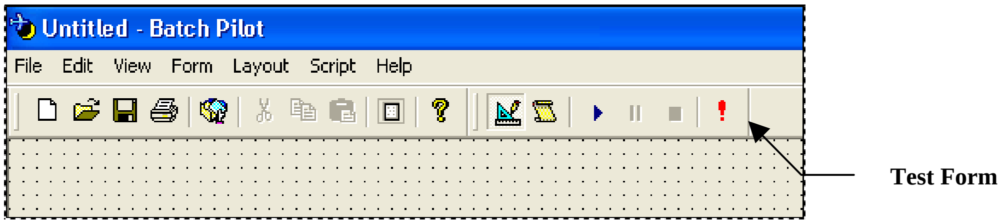
```
Sub InitCtrls
   ctrlLable1.Visible=True
   ctrlComment.Visible=True
        ↓
End Sub

Pilot.BatchStop()
Pilot.LoadObject(Comment)
Pilot.BatchContinue(4)
```

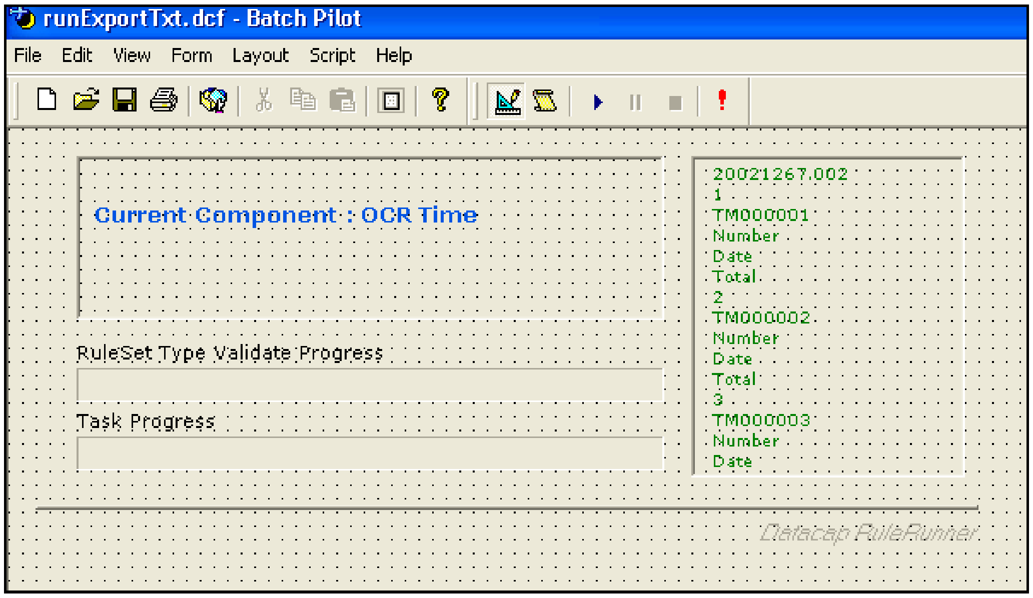
2. The script also has to apply the **ExportDB** rule that will update the Export database with the **Comment** field's value. ***Important!*** This happens automatically if the Task Definitions employs the dialog based on the *RuleRunner* ***setup*** form (Page 18).

How to Test a Form

After you open a Datacap form (.dcf) in the ***Batch Pilot Window***, you can test the form and its script just by clicking on the **Test Form** icon in the **Form** toolbar:



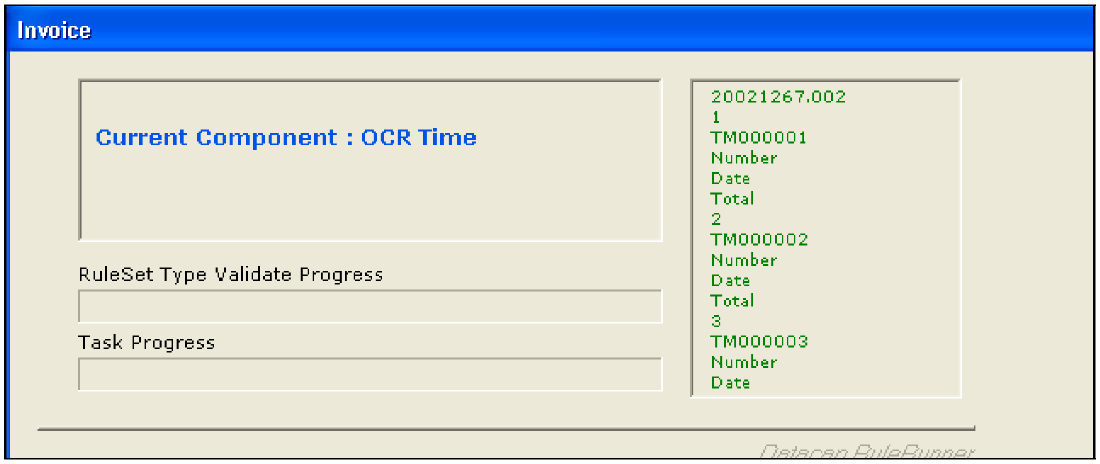
Below, the new **runExportTxt.dcf** form is open in the ***Batch Pilot Window –*** in *Design* mode:


runExportTxt.dcf - Batch Pilot
File Edit View Form Layout Script Help
Current Component : OCR Time
RuleSet Type Validate Progress
Task Progress
20021267.002
1
TM000001
Number
Date
Total
2
TM000002
Number
Date
Total
3
TM000003
Number
Date
Datacap RuleRunner

runExportTxt.dcf

If everything is OK, a single click on the **Test Form** icon will result in the form you see on the next page. ***Important!*** This will be the structure of the form when it becomes a task's ***setup*** or ***runtime*** dialog.

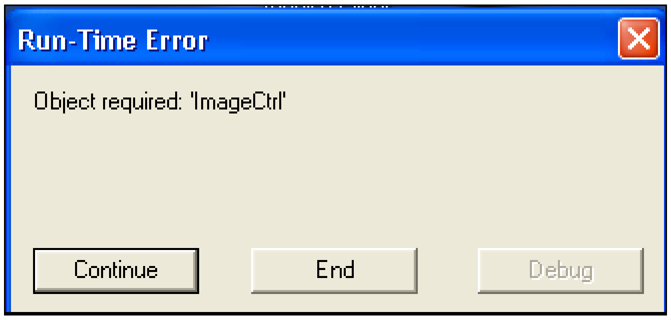
✓ Note the significantly different contents of the Title Bar.



```
20021267.002
1
TM000001
Number
Date
Total
2
TM000002
Number
Date
Total
3
TM000003
Number
Date
```

runExportTxt dialog

If there is a problem with the form, its controls or its script, you will receive an Error Message similar to this:



Object required: 'ImageCtrl'

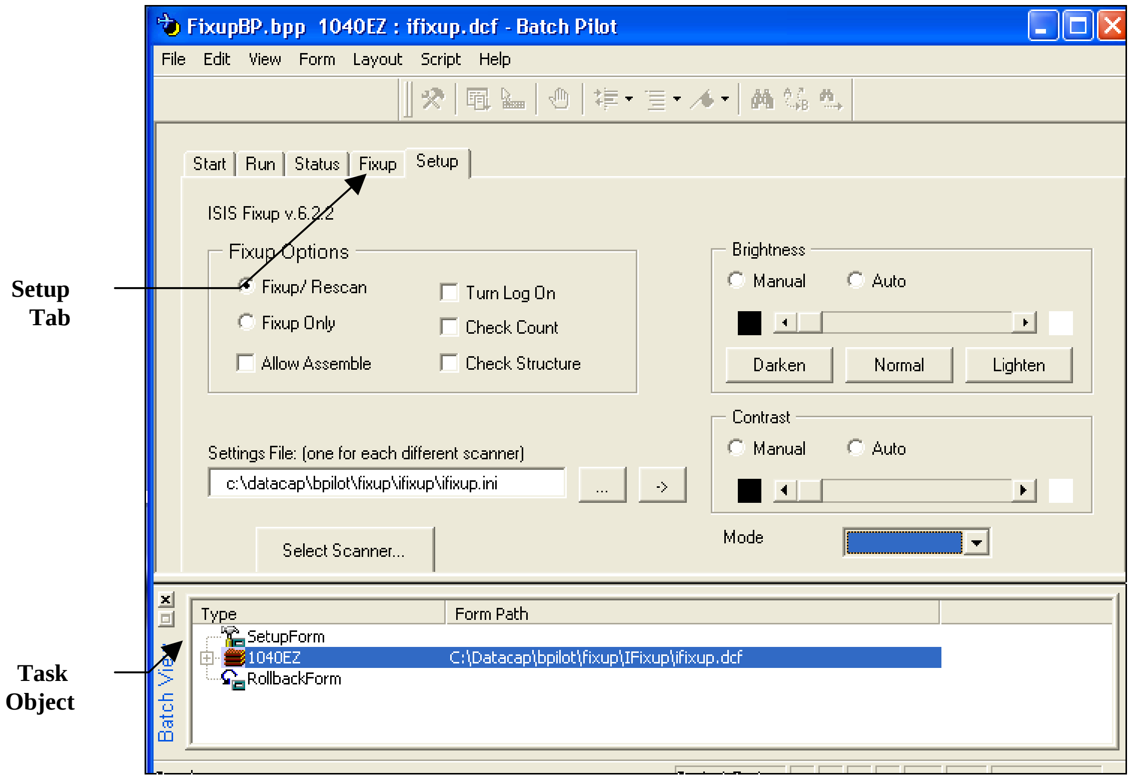
Form Test – Runtime Error

Multi-tab Forms

Most Task Projects (.bpp) have separate ***setup*** and ***runtime*** forms. These are **stock** forms (.dcf) that you'll find in the applicable sub-folder of the **Datacap** directory's **BPilot** folder.

The various Scan and FixUp forms, however, are **Multi-tab Forms**. A Multi-tab Form has multiple tabs: although an individual tab has ***setup*** or ***runtime*** attributes, the form itself is bound to a Task Project's **Batch** object in the ***Batch Pilot Window's* Setup Tree.**

✓ ***Alert!*** This connection can be confusing just because a single ***setup*** form is usually bound to the **SetupForm** type (below), while its corresponding ***runtime*** form is bound separately to the **Batch** object or to a **Page** object in the case of a ***Data Entry*** panel.


Setup Tab
Task Object
FixupBP.bpp 1040EZ : ifixup.dcf - Batch Pilot
File Edit View Form Layout Script Help
Start Run Status Fixup Setup
Fixup Options
Brightness
Darken Normal Lighten
Contrast
Select Scanner...
Batch View

ISIS Fixup v.6.2.2

Fixup Options: (●) Fixup/ Rescan, ( ) Fixup Only, ☐ Allow Assemble, ☐ Turn Log On, ☐ Check Count, ☐ Check Structure

Brightness: ( ) Manual ( ) Auto

Contrast: ( ) Manual ( ) Auto

Settings File: (one for each different scanner)
c:\datacap\bpilot\fixup\ifixup\ifixup.ini

Mode

Type	Form Path
SetupForm	
1040EZ	C:\Datacap\bpilot\fixup\IFixup\ifixup.dcf
RollbackForm	

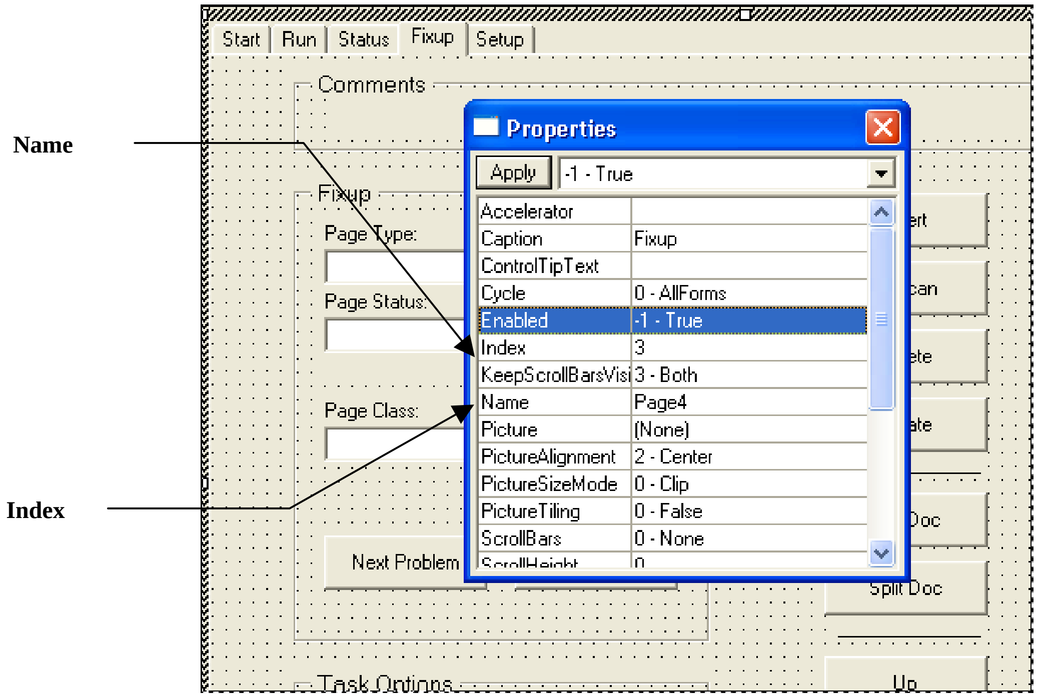
iFixUp Multi-tab Form – *Setup tab*

The `[General]` sector of a typical Multi-tab Form's code uses **constants** to set up a *zero-based* index of the form's tabs (and, by extension, its dialogs) with the following syntax (using the iFixUp Multi-tab Form as an example):

```
'property sheet page indexes
pStart=0
pRun=1
pStatus=2
```

```
pFixUp=3
pSetup=4
```

✓ However, values of a page's **Name** and **Index** properties (below) are ***not*** identical.



-1 - True

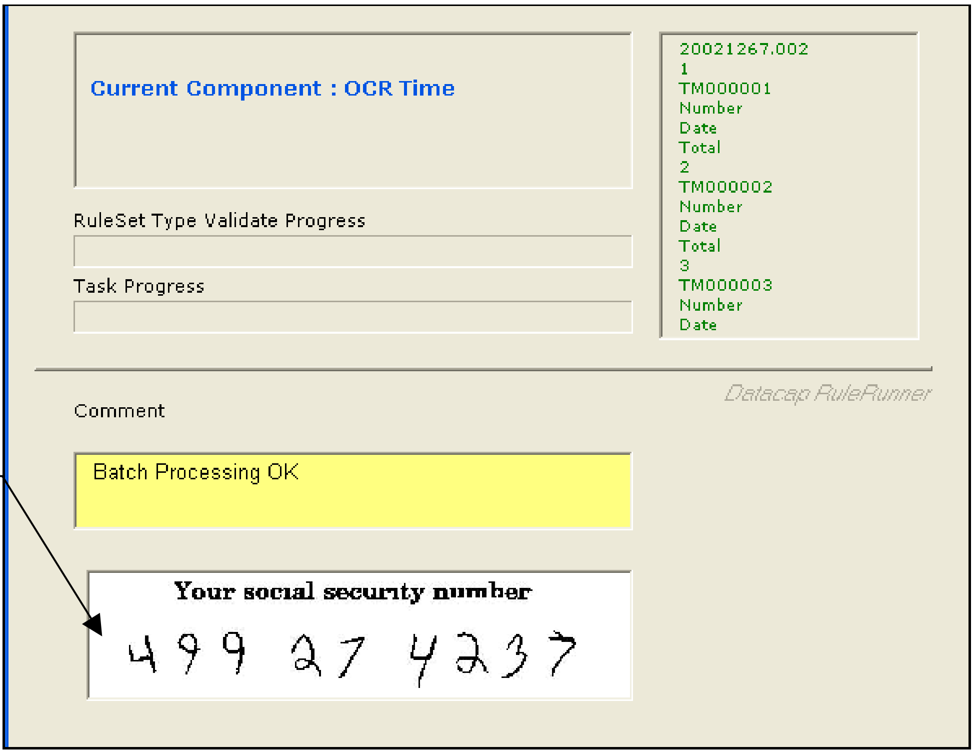
Accelerator	
Caption	Fixup
ControlTipText	
Cycle	0 - AllForms
Enabled	-1 - True
Index	3
KeepScrollBarsVisi	3 - Both
Name	Page4
Picture	(None)
PictureAlignment	2 - Center
PictureSizeMode	0 - Clip
PictureTiling	0 - False
ScrollBars	0 - None
ScrollHeight	0

iFIxUp Form – *Fixup tab*

Images

As Chapter 2 explains, the ***Batch Pilot Window*** can devote an entire section to images: after you open a form in *Design* mode, select **Image View** from the **View** menu.

You can use the standard **Image** control in the **Toolbox** to place an image directly on a form. Below, the revised ***runtime*** Export form now has a field that displays the ***image*** of the taxpayer's Social Security Number as it exports the data in each ***source*** page.



Current Component : OCR Time

RuleSet Type Validate Progress

Task Progress

```
20021267.002
1
TM000001
Number
Date
Total
2
TM000002
Number
Date
Total
3
TM000003
Number
Date
```

Datacap RuleRunner

Comment

Batch Processing OK

Your social security number

499 27 4237

Export Task – *Runtime Form*

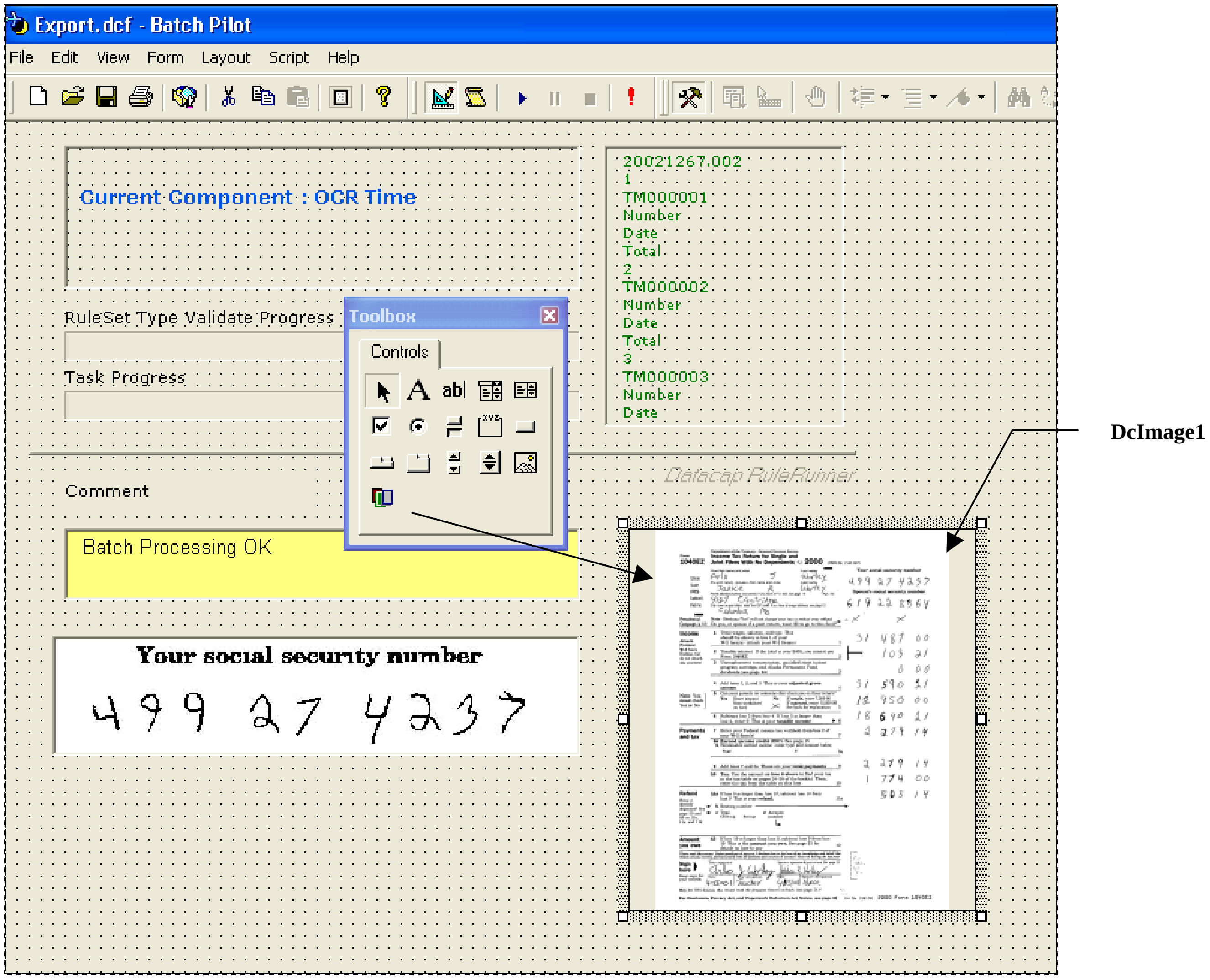
The image above belongs exclusively to the new **Export.dcf** form (Page 30).

But suppose you would like to display the image of the ***entire*** page in the **Image View** area of the ***Batch Pilot Window*** when the Export task runs?

✓ Coaxing an image into this area requires a little skill - and some experience with the **DCImage** control. This control is ***not*** part of the standard **Development Toolbox**: to add the control, follow the instructions on Page 23.

The statement below defines a relationship between *Batch Pilot's* **Image** control and its **DCImage** control. In this example, the **FileName** property of the **DCImage1** control object is assigned to the **Image Control** object.

```
ImageCtrl.FileName=DcImage1.FileName
```



Batch Pilot Window – DCImage Control

In this example,

- The **FileName** property of the **DCImage1** control has designated a specific image.
- The file's name and path is assigned to the **FileName** property of the ***Batch Pilot Window's*** **ImageCtrl** control.
- As a result, if the window's **Image View** area is open during processing (Chapter 2), the full image will appear in this area.

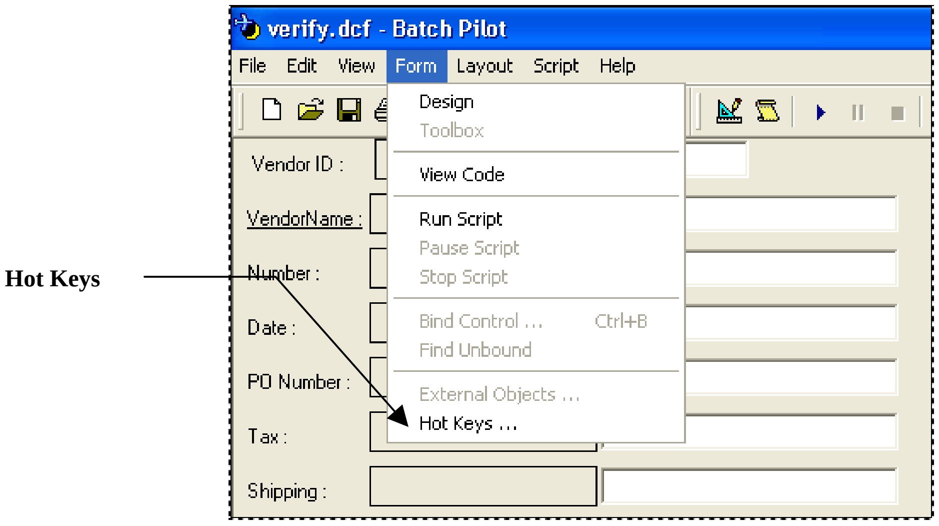
✓ Don't hesitate to experiment freely with existing *Batch Pilot* forms of the *1040EZ* application – and to assemble new forms.

Hot Keys

The ***Data Entry*** panels of ***pre-configured*** applications such as *1040EZ* or *Invoice* offer **hot keys** that an operator can use to move more efficiently through the panel as he or she reviews and, if necessary, corrects values in the panel.

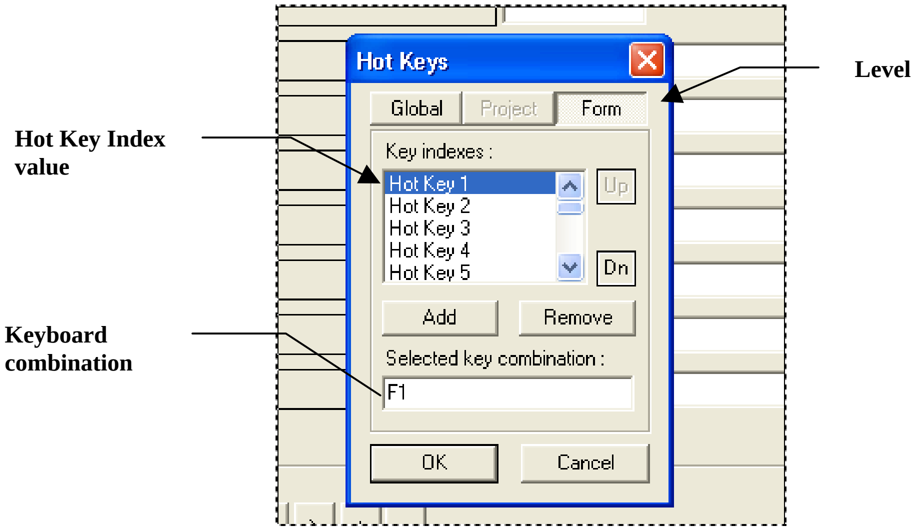
These hot keys belong to a specific ***runtime*** form such as the *1040EZ* application's **verify.dcf** form. To take a look, use *Batch Pilot* to access the application's **Process** directory and to open the form.

Then, select **Hot Keys** from the ***Batch Pilot Window's*** **Form** menu:



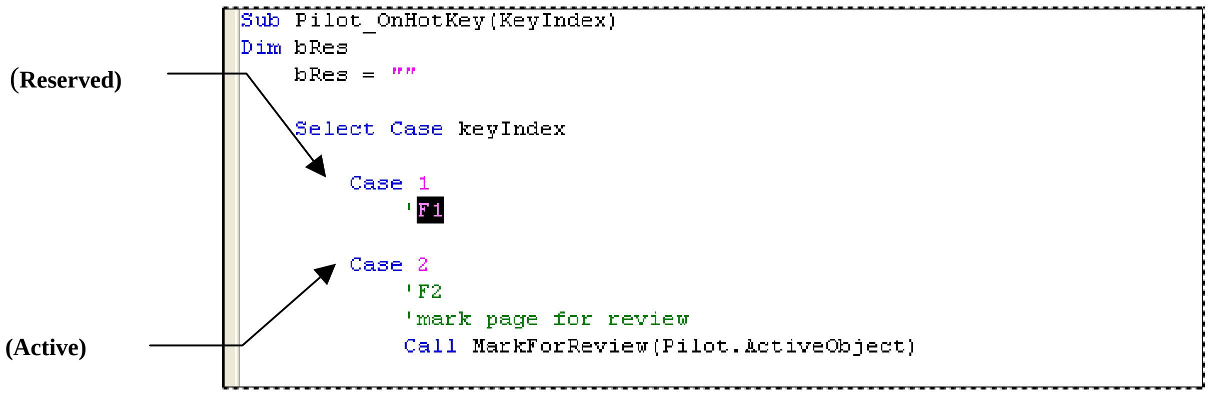
Batch Pilot Window – *verify.dcf form*

The ***Hot Keys*** dialog lists *one-based* index values for hot keys at the **form** level, and the keyboard combinations that launch the resulting activities. This dialog does not, however, tell you what a hot key does.



☛ For an explanation of a hot key's response:

- Select **View Code** from the **Form** menu;
- Switch to *Full Module View*;
- Click on the **Find** icon and conduct a search for "F1" (in this example);
- Review the code that appears on your screen.



```
Sub Pilot_OnHotKey(KeyIndex)
Dim bRes
    bRes = ""

    Select Case keyIndex

        Case 1
            'F1

        Case 2
            'F2
            'mark page for review
            Call MarkForReview(Pilot.ActiveObject)
```

How to Define a Hot Key

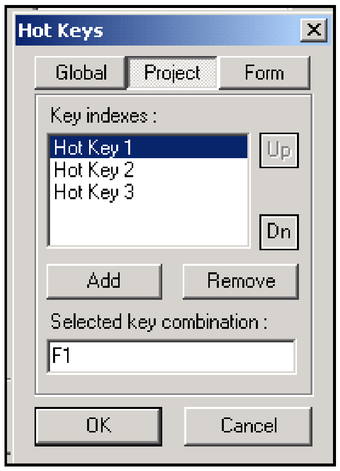
You can use *Batch Pilot's* **Hot Key** feature to define hot keys for a Task Project (.bpp), and hot keys for a form (.dcf).

✓ A **Hot Key Sequence** – *Ctrl + Alt + 4*, for example – which you assign to a hot key at the project level is available to ***any*** form bound to the project.

✓ A **Hot Key Sequence** – F3, perhaps – which you assign to a form is only available to that form.

A *Batch Pilot* hot key has two properties:

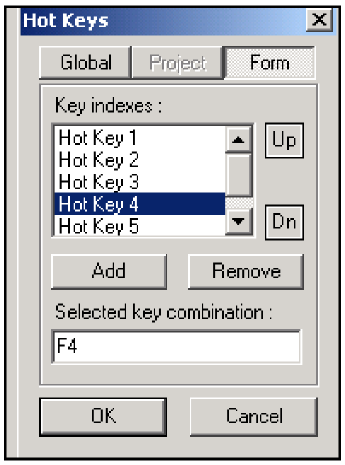
- **Index subscript.** The Hot Key Definition process on the next page assigns a *one-based **Hot Key Index*** subscript to the hot key. The index covers hot keys at the project level or the form level, but not both.
- **Sequence.** This property specifies the key or key sequenced on a keyboard that comprises the hot key.



Key indexes :
Hot Key 1
Hot Key 2
Hot Key 3

Selected key combination :
F1

Project Hot Keys



Key indexes :
Hot Key 1
Hot Key 2
Hot Key 3
Hot Key 4
Hot Key 5

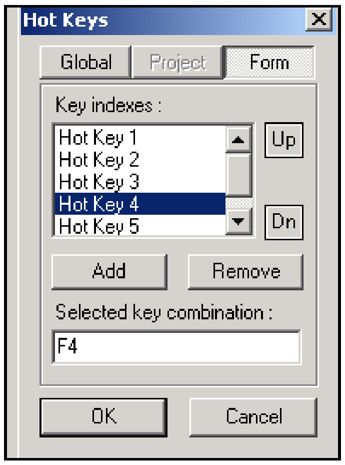
Selected key combination :
F4

Form Hot Keys

Although you can define ***Global*** hot keys that are available to all *Batch Pilot* projects and forms, we ***do not recommend*** this approach because it restricts your ability to compose custom hot keys for interactive Task Projects, such as those involved with Data Entry and Verification.

To define a hot key:

Step	Action
1.	Open *Batch Pilot.*
2.	Use the ***Batch Pilot Window's* File** menu to open a form.
3.	Select **Hot Keys** from the **Form** menu to access the ***Hot Keys*** dialog. (Note that the *Project* tab is not available in this example.)



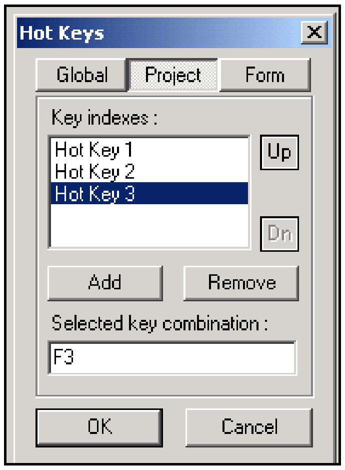
Key indexes :
Hot Key 1
Hot Key 2
Hot Key 3
Hot Key 4
Hot Key 5

Selected key combination :
F4

4.	Enter the new hot key's sequence in the **Selected Key Combination** field.
5.	Press the Add button; *Batch Pilot* will add the next available subscript to the **Key Indexes** list.

To Define a Hot Key (continued)

Step	Action
6.	Repeat Step #4 and Step #5 for any other hot keys. ***Remember:*** These hot keys are for the form you're working with only.
7.	Click on the OK button to close the ***Hot Keys*** dialog.
8.	Open the Task Project that will assimilate the form.
9.	Bind the form to the project (Page 3 and Chapter 4).
10.	Again, select **Hot Keys** from the **Form** menu to access the ***Hot Keys*** dialog:

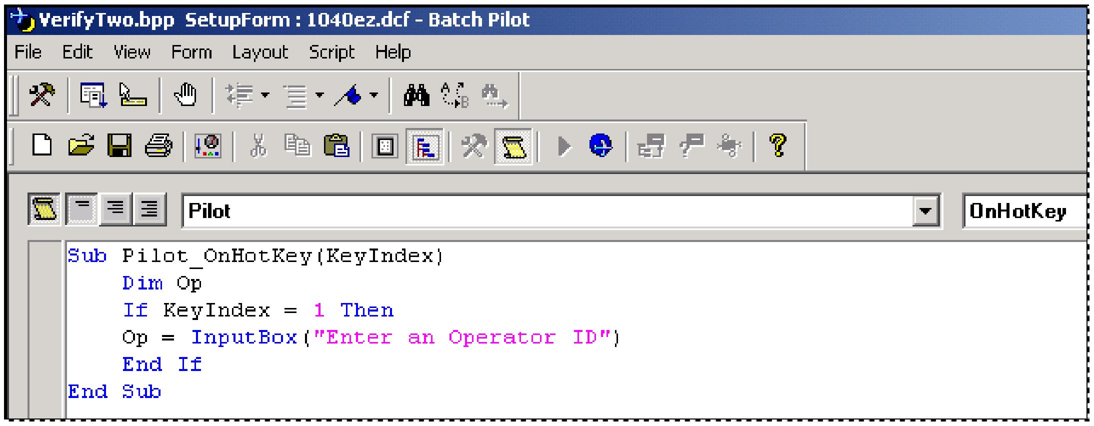


Here, the ***project's*** three hot keys – *Hot Key 1*, *Hot Key 2* and *Hot Key 3* – coincide with the ***form's*** first three hot keys. If there had been a conflict, *Batch Pilot* would alert you immediately and wait for you to resolve the problem.

✓ Any additional hot keys you assign to the project are instantly available to ***any*** form you bind to the project as long as there is no conflict. At the same time, the properties of a **stock** form may well include a set of pre-defined hot keys; when you bind the form to the project, conflict is almost inevitable.

Because defining new hot keys is such a simple procedure, removing a form's existing hot keys is often a worthwhile first step.

☛ The script you prepare for the **Pilot** object's **OnHotKey** event determines the responses of the individual hot keys.


VerifyTwo.bpp SetupForm : 1040ez.dcf - Batch Pilot
File Edit View Form Layout Script Help
Pilot
OnHotKey

```
Sub Pilot_OnHotKey(KeyIndex)
    Dim Op
    If KeyIndex = 1 Then
    Op = InputBox("Enter an Operator ID")
    End If
End Sub
```

Chapter 5 lists the pre-defined Hot Keys used by an application's ***Data Entry*** forms.