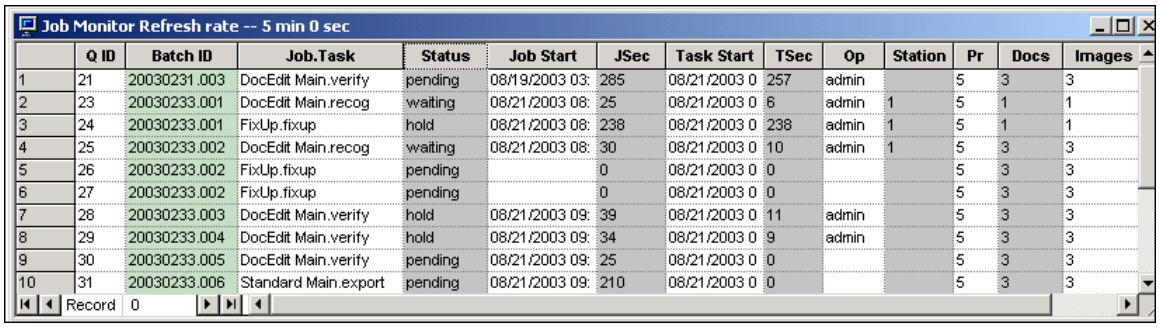
Enhanced Notification Utility

The *Enhanced Notification Utility (ENU)* applies rules you define to solicit information about the current state of your application, and e-mail the data directly to yourself and others on an *ad hoc* basis or according to a schedule you set up.

This documentation shows you how to configure the utility as a component of your application; how to define ENU rules; and how to run the utility in an *ad hoc* or *scheduled* mode. The documentation covers these topics:

Introduction

The ***Job Monitor*** is a ready source of data about the batches in the processing queues of your application's jobs and tasks. The illustration below shows the detail in just a few rows in the Batch Information Table of the *1040EZ* training application's ***Job Monitor.*** (For a thorough examination of all facets of the ***Job Monitor***, see Chapter 6 of the *Taskmaster Windows and Dialogs Reference.)*


Job Monitor Refresh rate -- 5 min 0 sec

	Q ID	Batch ID	Job.Task	Status	Job Start	JSec	Task Start	TSec	Op	Station	Pr	Docs	Images
1	21	20030231.003	DocEdit Main.verify	pending	08/19/2003 03:	285	08/21/2003 0	257	admin		5	3	3
2	23	20030233.001	DocEdit Main.recog	waiting	08/21/2003 08:	25	08/21/2003 0	6	admin	1	5	1	1
3	24	20030233.001	FixUp.fixup	hold	08/21/2003 08:	238	08/21/2003 0	238	admin	1	5	1	1
4	25	20030233.002	DocEdit Main.recog	waiting	08/21/2003 08:	30	08/21/2003 0	10	admin	1	5	3	3
5	26	20030233.002	FixUp.fixup	pending		0	08/21/2003 0	0			5	3	3
6	27	20030233.002	FixUp.fixup	pending		0	08/21/2003 0	0			5	3	3
7	28	20030233.003	DocEdit Main.verify	hold	08/21/2003 09:	39	08/21/2003 0	11	admin		5	3	3
8	29	20030233.004	DocEdit Main.verify	hold	08/21/2003 09:	34	08/21/2003 0	9	admin		5	3	3
9	30	20030233.005	DocEdit Main.verify	pending	08/21/2003 09:	25	08/21/2003 0	0			5	3	3
10	31	20030233.006	Standard Main.export	pending	08/21/2003 09:	210	08/21/2003 0	0			5	3	3

Record 0

Job Monitor – 1040EZ Application

Customarily, this processing data is available to you and other Administrative personnel when the application is up and running, and when you have a moment free to access and interrogate the ***Job Monitor***.

The *Enhanced Notification Utility* supplies part or all of the same information but in a different medium and format – regularly, according to a schedule you set up, or upon demand.

The illustration of the *ENU Batch Manager* on the following page highlights the elements of an ENU **rule** that resulted in the **e-mail** on Page 4.

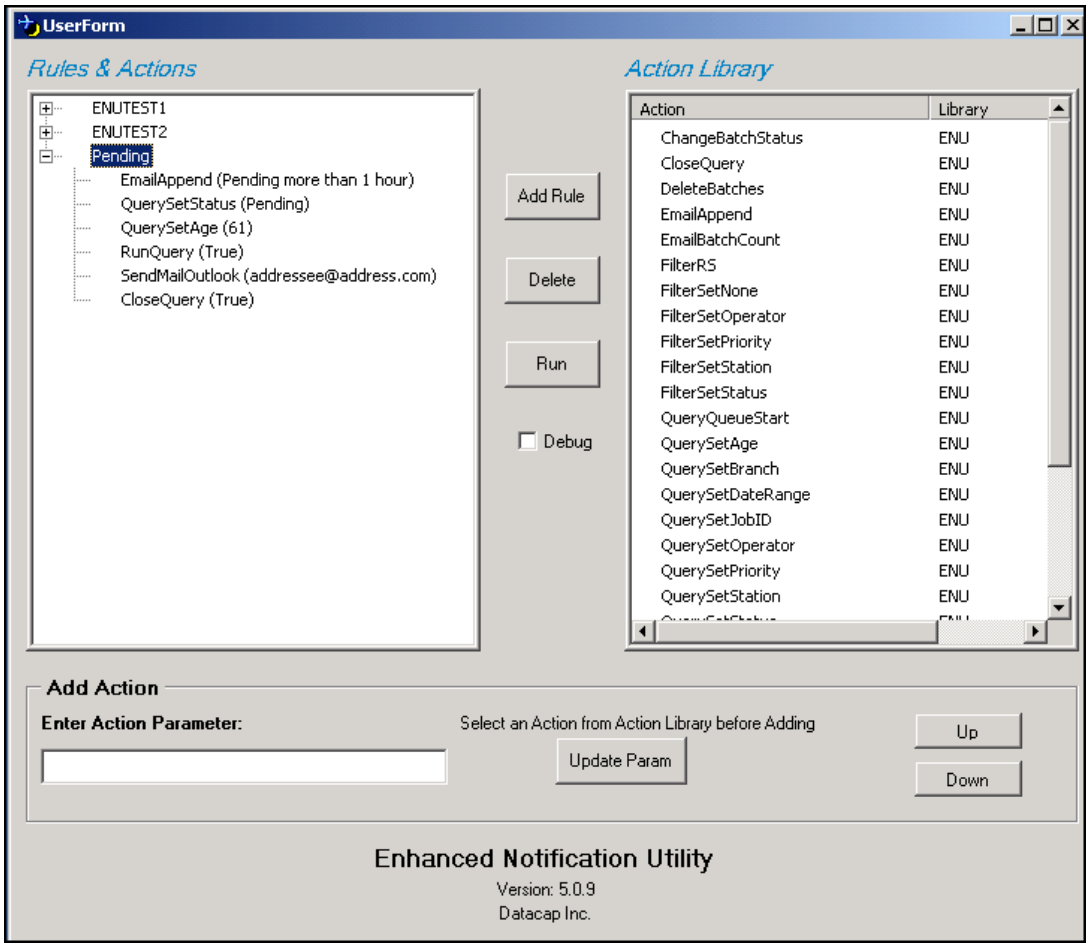
The goal of the *Pending* rule is to report on batches with a *Pending* processing status ***and*** an elapsed processing time greater than a hour.

The rule consists of **actions** drawn from the inventory on the right. Almost all actions need a **parameter** – a value you enter in the **Enter Action Parameter** field.

If you highlight the rule and click on the **Run** button, the rule carries out the actions, and forwards the data you've asked for as an e-mail communication.

The **Up** and **Down** buttons move an action one step above or below its current position in the rule.

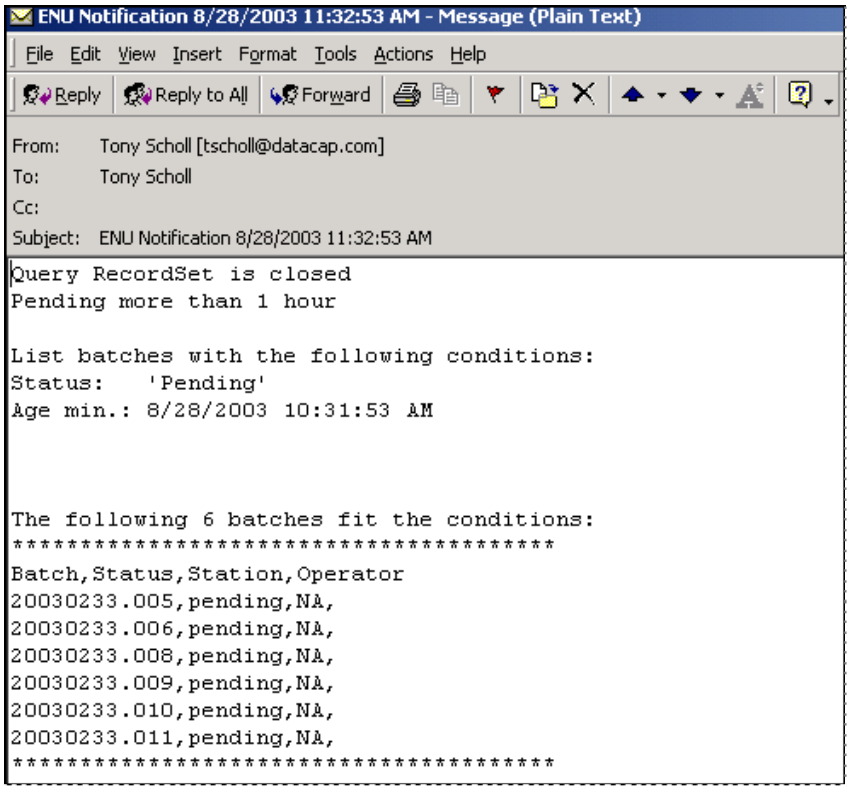
The **Update Param** button modifies a highlighted action's parameter; if you highlight an action in the list on the right, this button becomes the **Add Action** button.



ENU Batch Manager

The *Pending* rule you see above employs the following actions:

- `EmailAppend(Pending more than 1 hour)` adds a title line to the e-mail – see the example on the next page.
- `QuerySetStatus(Pending)` asks for a list of batches with a *Pending* status.
- `QuerySetAge(61)` limits the list to batches which have been in the current state for more than one hour.
- `RunQuery(True)` carries out the rule's **Query** actions.
- `SendMailOutlook(addressee@address.com)` uses *Microsoft Outlook* to e-mail the rule's results to the e-mail address entered as a parameter.



Pending Rule - *e-mail*

The rule's communication has these sections:

E-Mail: From and To. In this opening example, the individual who applied the rule sent the batch data to himself.

E-Mail: Subject. This is a standard ENU declaration.

Communication Status. The e-mail's first line indicates that ENU processed the rule successfully. If there are problems, Error Messages appear, and you can check the contents of the ENU log (Page 24).

Communication Title. This is a brief but ***very helpful*** description of the rule and the data you expect to be in the e-mail.

Selection Criteria. This section summarizes the nature of the rule's **SetQuery** actions.

Results. Beneath a set of column headings, the e-mail lists those batches which satisfy the selection criteria.

Upcoming sections show you how to install and configure the *Enhanced Notification Utility*, and how to define and run rules in an *ad hoc* or *scheduled* mode. The section on Page 12 explains each ENU action.

ENU Installation and Configuration

The *Enhanced Notification Utility* is supplementary software that uses *Batch Pilot* technology to report on *Taskmaster* activity. Communications take the form of e-mails to one or more addressees; ENU scheduling (if you choose this reporting mode) employs the *Windows Task Scheduler*.

✓ As a result, ENU can run ***only*** if your **Datacap** application's environment includes:

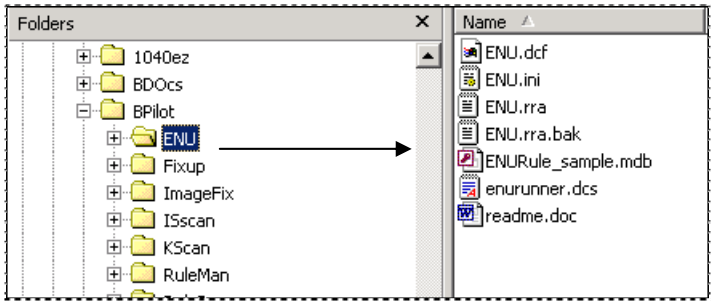
Taskmaster 5 or above for workflow development and operation.

Batch Pilot for ENU rule definition and application.

Installing ENU Files

Datacap Installation places these files in the **ENU** folder of the **Batch Pilot** sub-directory:

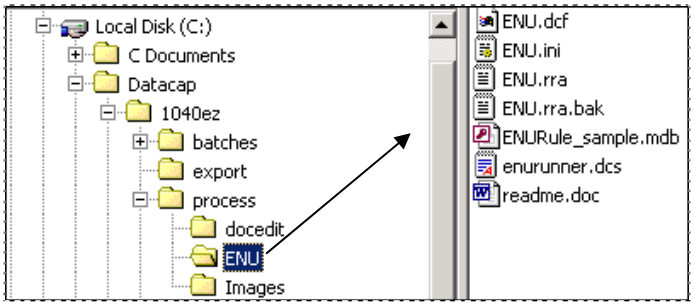
- **ENU.dcf** is the *Batch Manager* pictured on Page 3.
- **ENU.ini** is the utility's Settings file. (For a complete explanation, see Page 7).
- **ENURule_sample.mdb** is an *Access* database that will maintain details of the rules you define. (This is a default name you'll probably change when you link the database to your application – Page 6. Alternatvely, you can use a SQL Server or Oracle database. For details, see Page XX.)
- **readme.doc** is a brief summary of the configuration steps.
- **enu.rra** is the library of ENU actions (Page 12).
- **enurunner.dcs** is the script that responds to your rules by retrieving *Taskmaster* data and forwarding it to an e-mail's addressee(s).



ENU Files

☛ Installation consists of one simple step: copy these files ***from*** *Batch Pilot's* **ENU** folder ***to*** a folder within your application's **Datacap** sub-directory. Maybe you'll place these files in the application's **Process** folder. Often, however, that folder has many files; for simplicity, you may want to add an **ENU** folder to hold the utility's files.

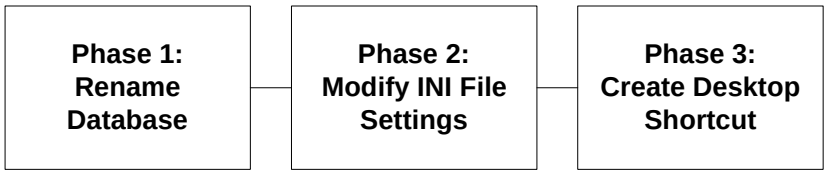
In this example, the *1040EZ* Administrator has taken the latter approach:



ENU files – *1040EZ Application*

ENU Configuration

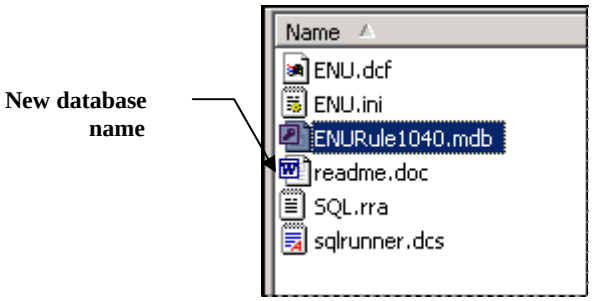
Configuring your *Enhanced Notification Utility* is almost as easy, and involves these steps:



Phase 1: Rename the Rules Database

Although this preliminary step is *optional*, it can add measurably to the configuration's overall clarity.

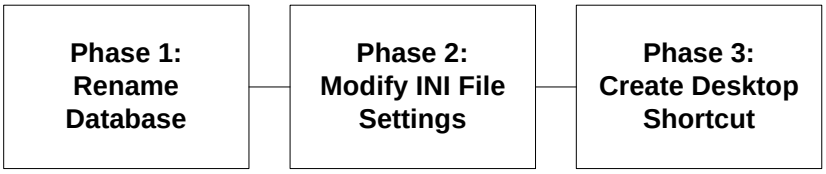
Here, the *1040EZ* Administrator has changed the default name – **ENURule_sample.mdb** to **ENURule1040.mdb**. This small but important change identifies the database with the application and with the utility, rather than with the utility alone.



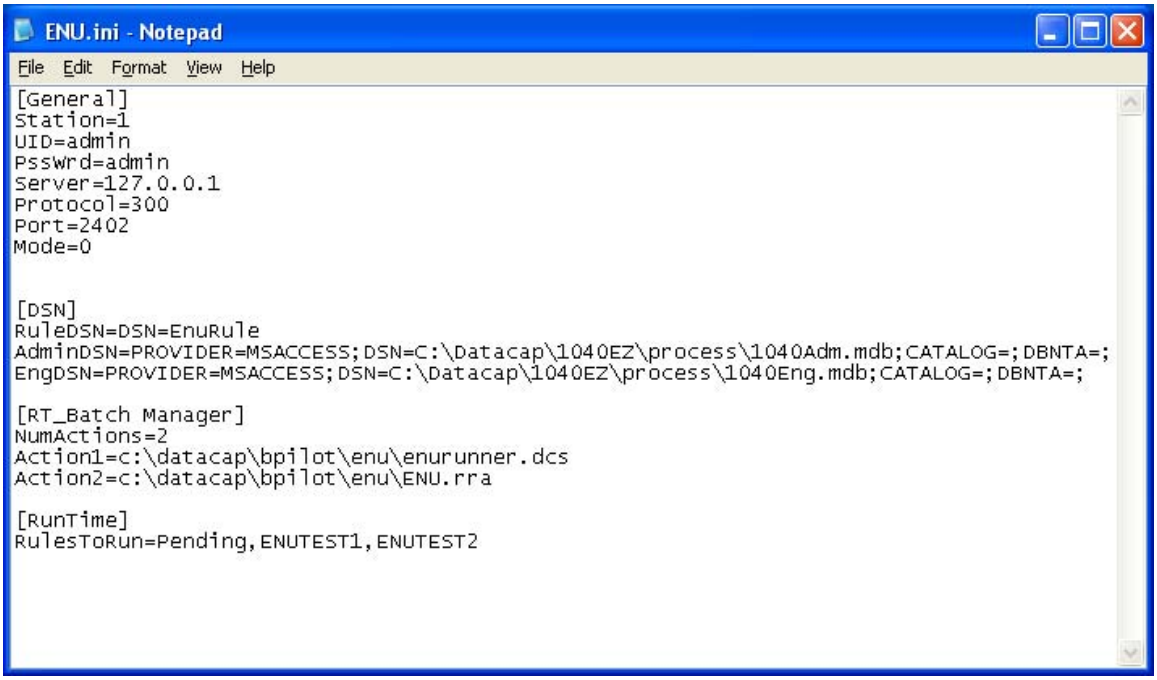
✓ To change the name, use your Windows Explorer to locate the file. Right-click on the file's original name, select **Rename** from the options, and enter the new title.

Phase 2: Update the ENU Settings File (.ini)

In Phase 2, you'll modify the specifications of most items in a short Settings file:



This is one of the files you added to your application during Installation (Page 5). When you open the file with a tool such as *Notepad*, the file and its ***default*** settings will look like this:



ENU.ini

The table on the next page reviews the nature of each setting, and explains the changes you're likely to make.

ENU Settings File (.ini)

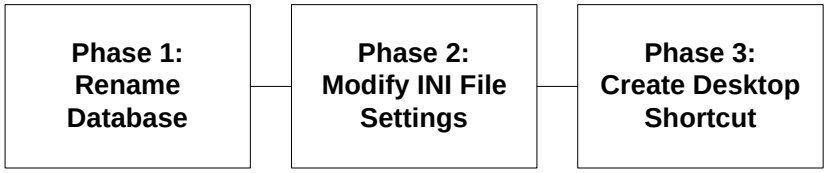
Section/Setting	Description
[General]	The settings in this section affect ENU access and operations.
Station=1	The first of the ENU Security settings. This is the Station ID you enter when signing on to your Taskmaster Client. "1" is the ***default*** value; in most cases, you'll use this value.
UID=admin	The User ID your Enhanced Notification Utility will use when logging on to Taskmaster Server. Again, this value will usually be the same as the User ID you enter when you open Taskmaster Client.
Psswrd=Admin	The User Password.
Server=127.0.0.1	The **I**nternet **P**rotocol address of your configuration's Taskmaster Server. ***Important!*** Taskmaster Server links the *Enhanced Notification Utility* to your application's Engine and Admin databases – even when your Taskmaster Client is not running.
Protocol=300	A number representing the server's Transmission Control Protocol – in this case, **TCP/IP**. ***Alert!*** Although Taskmaster Server can support other protocols, we ***strongly recommend*** this standard. If you are considering an alternative, please consult your Datacap Implementation specialist first.
Port=2042	A number designating Taskmaster Server's Communications port. This ID is assigned when you set up Taskmaster Server (see Chapter 3 of the *Taskmaster Administrator' Guide.)*.
Mode=0	A number indicating if ENU is to run in an *interactive* mode (0) or an *unattended* mode (1). When you are setting up the utility, assembling rules and testing ENU communications, this value is "0". If ENU is running behind-the-scenes in response to a schedule you've established (Page 26), the value is "1".

ENU Settings File (.ini)

Section/Setting	Description
[DSN]	The Connection Strings link the *Enhanced Notification Utility* to its Rules database - ***and*** to your application's Admin and Engine databases. The illustration of the ENU.INI file shows default Connection Strings linking ENU to the Admin and Engin databases of the *1040EZ* training application. When you set up the file, the Connection Strings will designate your databases. ***Alert!*** These are *OLE Microsoft Access* Connection Strings. **Datacap Taskmaster** also supports SQL Server and Oracle methods to access ENU. Appendix A in the Taskmaster Administrator's Guide lists the ENU Connection String syntax for each method.
[RT_Batch Manager]	This section identifies two key *Enhanced Notification Utility* files. Although you do not want to change the files' names, you will have to modify their pathways to reflect their locations in your application's directory hierarchy.
NumActions=2	Indicates the number of ENU operations files. Do ***not*** change this value.
Action1=c:\Datacap\ bpilot\ enu\ enurunner.dcs	The name and location of the script that applies the ENU rules you define (Page 11), then generates and forwards the resulting e-mails. Although you should ***not*** alter the script's name and extension, you do want to update the file's pathway. The setting for the *1040EZ* application is: Action1=c:\Datacap\1040EZ\Process\ enu\enurunner.dcs
Action2= c:\Datacap\ bpilot\ enu\ enu.rra	The name and location of the **Actions File** which contains the ENU actions (Page 12). Again, you should ***not*** alter the script's name and extension, but you do want to update the file's pathway.
[Runtime]	This specification identifies one or more rules that you have defined (Page 11) – rules that will be applied automatically if ENU is running in a *scheduled* mode (and the **Mode** value above is 1!)
RulesToRun=ENUTEST1, ENUTEST2	Two trial rules we've constructed to test your *scheduled* operations (Page 26).

Phase 3: Create a Desktop Shortcut

The closing phase of the Configuration process defines and places an ENU shortcut on your desktop:

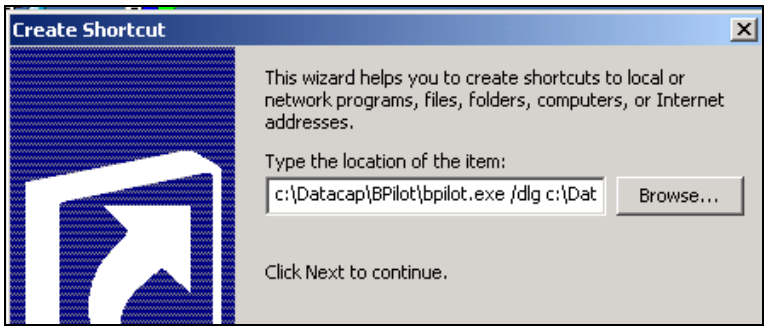


This phase involves the following steps:

Step	Action

1. Right-click on an open space in your desktop.

2. Select **New** and **Shortcut** from the options.



3. When the ***Create Shortcut*** dialog appears, enter the following data in the shortcut's command line, adhering strictly to the format:

```
C:\Datacap\Bpilot\Bpilot.exe /dlg C:\Datacap\
  1040EZ\Process\enu\enu.dcf -rs \enu.ini
```

Alert! Be sure to replace "1040EZ" references with the name of ***your*** application's directory.

4. Close the command line's dialog; double-click on the shortcut icon to open your application's ENU ***Batch Manager***.

Problems you may encounter:

Double-clicking on the **ENU** shortcut is a reliable test of the Configuration process.

Error messages will alert you immediately if there are problems with any stage. The most likely errors involve your definition of the specification of a Connection String in the ENU Settings file during Phase 2 (Page 7).

ENU Rule Definitions

An ENU rule consists of a series of ENU actions that run sequentially.

A typical rule uses the following action structure to generate and e-mail a communication:

EmailAppend(title)	Gives the communication a title. The title is usually the opening line of the e-mail (see the example on the next page.)
QuerySetBasis1(parameter)	The query's first selection level As an example, `QuerySetJobID(Main)` selects all batches being processed by tasks of the application's Main job.
QuerySetBasis2(parameter)	The query's second selection level. `QuerySetTaskID(Recog)` further limits the selection to batches awaiting processing by the Main job's Recognition task.
QuerySetBasis*N*(parameter)	The query's *n*th selection level. This **SetQuery** action might specify a parameter of a very different nature such as a Station or Operator. However, because of this action's placement in the rule, it applies ***only*** to those batches selected by the preceding actions.
RunQuery(True)	Runs the query according to the specifications of the **QuerySet** actions and **FilterSet** actions.
SendMailOutlook/CDONTS (e-mail addressees)	Prepares and formats an e-mail, adds the rule's batch information, and forwards the communication to the recipient(s) you name.

ENU Actions

The table below describes each ENU action. The section on Page 18 shows you how to set up a rule, provide it with actions, and test the rule ***and*** its actions.

ENU Action	Description	Example
Rule Management Action	These actions opens and runs rule's query procedures.	

`RunQuery`	Initiates the rule's sequence of **Query** actions.
	
	Parameter: *Boolean* value indicating *True* or *False*.
		`QuerySetJobID(Main)`
		`QuerySetTaskID(Recog)`
		`RunQuery(True)`
		
		Here, the **RunQuery** action carries out two **QuerySet** actions.

`CloseQuery`	An optional action that closes the current recordset. **Parameter: *???***	`RunQuery(True)` **`CloseQuery`**
Query Actions	These actions locate and retrieve data about a workflow's active batches. ***Alert!*** Multiple **Query** actions filter the selection of batches sequentially: the batches which remain must satisfy the criteria of all actions.	

QuerySetJobID	Locates those batches in the processing queues of one or more jobs.
	
	Parameter: *String* value(s) of one or more Job IDs.
	
	Alert! Separate multiple Job IDs with commas.
		`QuerySetJobID(Main)`
		
		This action above returns information about all batches being processed by the tasks of a workflow's "Main" job. The action below provides the communication with information about the batches of two jobs: *Main* and *FixUp*.
		
		`QuerySetJobID(Main,`
		`FixUp)`

ENU Actions (continued)

ENU Action	Description	Example
QuerySetTaskID	Retrieves information about batches in the processing queue of a particular task. **Parameter:** *String* value(s) of one or more Task IDs. Be sure to separate multiple Task IDs with commas.	`QuerySetTaskID(Verify)` If a rule contains this action without a preceding **QuerySetJobID** action, the communication will list all batches in line for the Verify tasks of all jobs.
QuerySetStatus	Retrieves information about batches with a specific processing status. **Parameter:** *String* value(s) of one or more Batch Statuses. Separate multiple Statuses with commas.	`QuerySetTaskID(Verify)` **`QuerySetStatus(Pending)`** This action will list any batch with a *Pending* status **if** it is in line for the Verify task.
QuerySetPriority	Retrieves information about batches assigned a specific processing priority. **Parameter:** *Numeric* value(s) of one or more **Priority** property. Multiple priorities are separated by commas.	`QuerySetPriority(8)` **Priority** is a property of a Job Definition and is passed on to the job's tasks and from the tasks to the batches. ("10" is *Low*, "1" is *High*, and "5" is *Average* ***and*** the default.) Typically, this action looks for exceptional priority values, and follows JobID and Task ID queries.
QuerySetBranch	Lists batches that have generated sub-batches *n* times during their processing histories. **Parameter:** *Numeric* value of *n* – how many times a parent batch has spawned children.	`QuerySetBranch(3)` This action will report on any batch that has been subject to splitting mechanisms exactly three times.

ENU Actions (continued)

ENU Action	Description	Example
QuerySetAge	Locates batches that have been in a processing queue for ***at least*** *n* minutes. **Parameter:** *Numeric* value of the minimum minutes.	`QueryQueueStart(False)` **`QuerySetAge(20)`** By itself, the action lists all batches that have been in the processing queues of ***any*** Job/Task combination for at least 20 minutes. Usually, however, this action is preceded by qualifying actions that limit the scope of the query to certain jobs and tasks perhaps, or statuses (see the example on Page 18).
`QueryQueueStart`	Determines the selection basis for subsequent **QuerySetDateRange** or **QuerySetAge** actions. **Parameters:** *Boolean* value indicating *True* or *False* (see the example.)	**`QueryQueueStart(True)`** `QuerySetAge(60)` In this example, all batches that the "current" task ***started*** processing at least one hour ago will be selected. If the parameter is *False*, the action will select any batches that the current task ***finished*** processing at least one hour ago. ***Alert!*** This setting applies to all subsequent uses of **QuerySetAge** or **QuerySetDate Range** in any rule – during the current run of the *Enhanced Notification Utility*. If this action is not called before a **QuerySetAge** action or a **QuerySetDate Range** action (below), the default value is *False.*

QuerySetDateRange	Specifies batches that have been in a processing queue from Date 1 ***through*** Date 2.
	
	Parameter: Two *Date* values separated by a comma.
		`QuerySetJobID(FixUp)`
		`QueryQueueStart(False)`
		`QuerySetDateRange`
		`(10/01/04, 10/06/04)`
		
		The pair above first selects all batches belonging to the FixUp job, then limits the selection to batches that entered the job's processing queue sometime during the Date Range.
		
		If a **QuerySetTaskID** action follows the **QuerySetJobID** action, the Date Range would apply to the batches being processed by the task as part of the job.

ENU Actions (continued)

ENU Action	Description	Example
QuerySetOperator	Identifies batches in line for processing by a particular operator. **Parameter:** The *string* value(s) of one or more Operator IDs. ***Be sure*** to separate multiple Operator IDs with commas.	`QuerySetOperator(sup1, verify1)` In the absence of qualifying actions, this action lists all batches that are the responsibility of a Supervisory operator (*Sup1*) and a Verification operator (*Verify1*).
QuerySetStation	Identifies batches in line for processing by a particular workstation. **Parameter:** The *string* value(s) of one or more Station IDs. Separate multiple Station IDs with commas.	`QuerySetStation(scan1, de1)` This action lists all batches that are the responsibility of the *scan1*and *de1* workstations.
Additional Actions	The first action operates on batches selected by the **Query** actions; the second assists with **Date** and **Time** parameters if your application uses a SQL Server database.	

ChangeBatchStatus	Changes the processing Status of batches selected by a rule's earlier actions.
	
	Parameter: The *String* value of the new Status.
		`QuerySetJobID(Main)`
		`QuerySetTaskID(Export)`
		`QuerySetStatus(Waiting)`
		`ChangeBatchStatus(Finished)`
		
		This sequence corrects a processing glitch in which batches processed by the Main job's Export task have been assigned *Waiting* rather than *Finished* as their processing status.

ENU Actions (continued)

ENU Action	Description	Example
SetSQLSeparator	The *Enhanced Notification Utility* assumes that your *Taskmaster* databases are Access databases. If, instead, you employ SQL Server databases, you ***must*** include this action before a rule's **QuerySetAge** or **QuerySetDateRange** action. **Parameter:** An apostrophe (‘). This is the separator that a SQL Server database places between *Date* values.	**`SetSQLSeparator(’)`** `QuerySetAge(180)` In this example, the opening action gives the second action the syntax it needs to compare *Date* values in a SQL database.
Communication Management Actions	The **Communication Management** actions organize a rule's data, place it in an e-mail, and address and send the e-mail.	

EmailAppend	Provides the communication with a title, and places the title at the beginning of the e-mail.
	
	Parameter: *String* value of the communication's title.
		`EmailAppend(Recog Batches with a Waiting Status)`
		`QuerySetTaskID(Recog)`
		`QuerySetStatus(Waiting)`
		
		The opening action in this sequence provides the e-mail with a Title Line that clearly identifies the nature and scope of the communication's data.

SendMailOutlook	Uses *MS Outlook* to prepare the formatted e-mail for distribution to the addressees you identify in the Batch Manager's **Parameters** field.
	
	Parameter: The e-mail address(s) of the individual (or individuals) who are to receive the ENU communication. ***Be sure*** to separate multiple addresses with commas.
		`<Query actions>`
		`RunQuery(True)`
		`SendMailOutlook (jack@hq.com,jill@hq.com)`
		`CloseQuery`
		
		This sequence prepares to distribute an *Outlook* e-mail to two recipients: Jack and Jill. However, it does not actually send the e-mails. Instead, it places them in your ***Outbox,*** where they remain until you use *Outlook's* Send/Receive mechanism to forward them.

ENU Actions (continued)

ENU Action	Description	Example

SendMailCDONTS	Uses software other than *MS Outlook* to prepare and distribute the e-mail.	`<Query actions>`
		`RunQuery(True)`
		`SendMailCDONTS(jack@hq.com)`
		`CloseQuery`

SendMailCDOSYS	Sends the in-progress e-mail via CDOSYS(Windows Server SMTP) Service) on Windows 2003 Server.
	
	Parameter: The *String* value of the e-mail address. Insert a semi-colon (;) to separate two or more e-mail addresses.
		`RunQuery(True)`
		`SendMailCDOSYS(jack@hq.com;`
		`larry@hq.com;lisa@hq.com)`
		`CloseQuery`

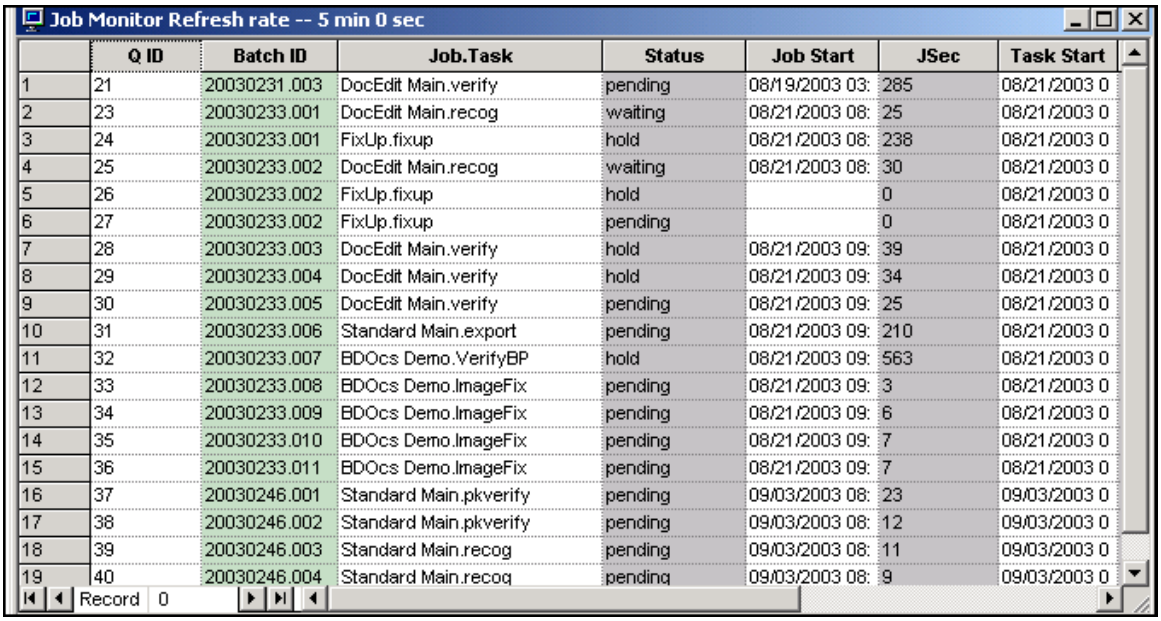
SetLogFile	*Inactive*	

How to Define and Test a Rule

The structure and tools of the ***ENU Batch Manager*** ensure that Rule Definition is a straightforward process: easy-to-read Error Messages and Log files help measurably as you formulate and test a rule and its actions.

✓ Before you assemble ENU rules for your own application, why not "experiment" with the *1040EZ* training application? *1040EZ* is nearly indestructible and encourages exploration of every sort! The discussions of ENU Installation and Configuration (Pages 5 and 6 respectively) use examples that show you how to set up an *Enhanced Notification Utility* for *1040EZ.* All that's left for you is to be sure that the *1040EZ* ***Job Monitor*** lists plenty of active batches, widely distributed among the application's jobs and tasks (see the illustration below.)

Alert! Before you sign on to Taskmaster Client for *1040EZ* (or your application), be sure Taskmaster Server is running and that the application's *ENU Batch Manager* is ***not*** running.



	Q ID	Batch ID	Job.Task	Status	Job Start	JSec	Task Start
1	21	20030231.003	DocEdit Main.verify	pending	08/19/2003 03:	285	08/21/2003 0
2	23	20030233.001	DocEdit Main.recog	waiting	08/21/2003 08:	25	08/21/2003 0
3	24	20030233.001	FixUp.fixup	hold	08/21/2003 08:	238	08/21/2003 0
4	25	20030233.002	DocEdit Main.recog	waiting	08/21/2003 08:	30	08/21/2003 0
5	26	20030233.002	FixUp.fixup	hold		0	08/21/2003 0
6	27	20030233.002	FixUp.fixup	pending		0	08/21/2003 0
7	28	20030233.003	DocEdit Main.verify	hold	08/21/2003 09:	39	08/21/2003 0
8	29	20030233.004	DocEdit Main.verify	hold	08/21/2003 09:	34	08/21/2003 0
9	30	20030233.005	DocEdit Main.verify	pending	08/21/2003 09:	25	08/21/2003 0
10	31	20030233.006	Standard Main.export	pending	08/21/2003 09:	210	08/21/2003 0
11	32	20030233.007	BDOcs Demo.VerifyBP	hold	08/21/2003 09:	563	08/21/2003 0
12	33	20030233.008	BDOcs Demo.ImageFix	pending	08/21/2003 09:	3	08/21/2003 0
13	34	20030233.009	BDOcs Demo.ImageFix	pending	08/21/2003 09:	6	08/21/2003 0
14	35	20030233.010	BDOcs Demo.ImageFix	pending	08/21/2003 09:	7	08/21/2003 0
15	36	20030233.011	BDOcs Demo.ImageFix	pending	08/21/2003 09:	7	08/21/2003 0
16	37	20030246.001	Standard Main.pkverify	pending	09/03/2003 08:	23	09/03/2003 0
17	38	20030246.002	Standard Main.pkverify	pending	09/03/2003 08:	12	09/03/2003 0
18	39	20030246.003	Standard Main.recog	pending	09/03/2003 08:	11	09/03/2003 0
19	40	20030246.004	Standard Main.recog	pending	09/03/2003 08:	9	09/03/2003 0

1040EZ Job Monitor – *Left Columns*

To put together a rule that assembles and communicates information about all batches waiting to be processed by the DocEdit Main **job's** Verify **task**, you could take the steps outlined on the next page.

To Construct and Run an ENU Rule

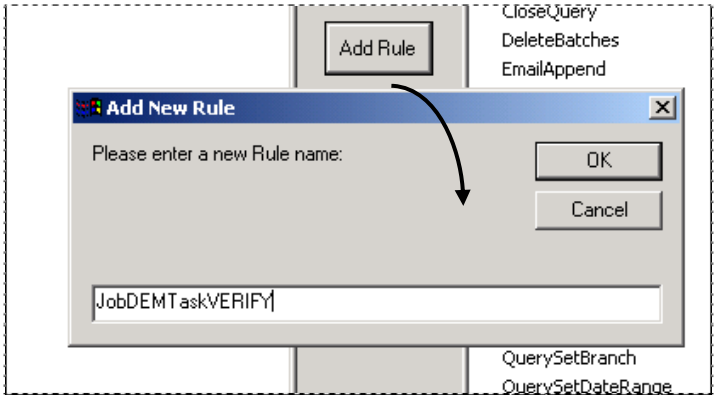
Step	Action

1. Be sure Taskmaster Server is running.

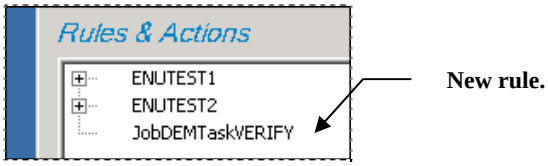
2. Close the application's Taskmaster Client (if it's open.)

3. Double-click on your desktop's **ENU** shortcut icon to access the *ENU Batch Manager* (illustrated on the next page.)

4. Press the Add Rule button. When the *Add New Rule* dialog appears, enter the rule's name in the empty text field and click on the OK button.

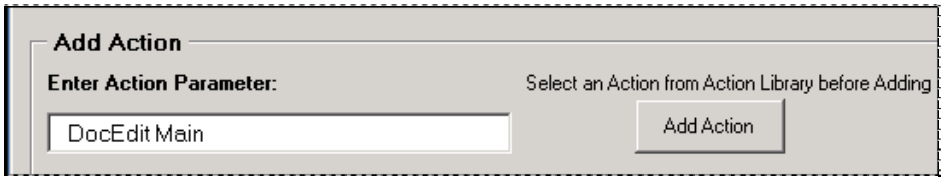


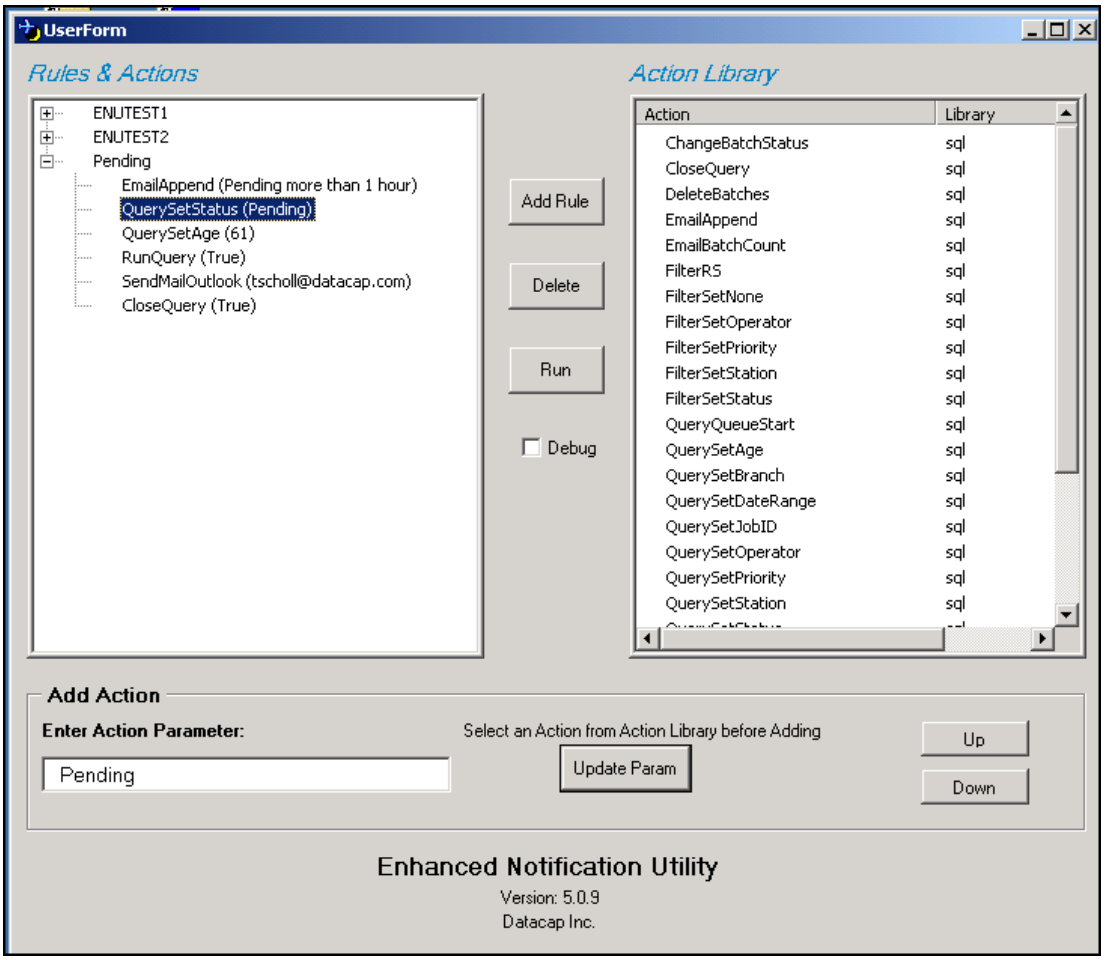
5. Be sure the *Batch Manager's* **Rules & Actions** sector has added the new rule's name to its list:



6. Select the rule's first action from the list of actions in the *Batch Manager's* **Action Library** sector on the right:

7. Enter the appropriate parameter(s) for this action in the **Add Action** sector's **Enter Action Parameter** field:



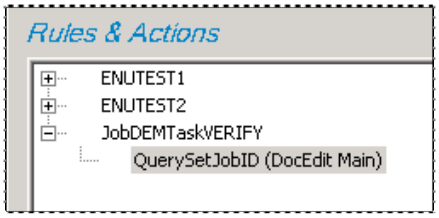


ENU Batch Manager

To Define a Rule (continued)

Step	Action

8. Click on the Add Action button. Check to be sure the action is now part of the rule:

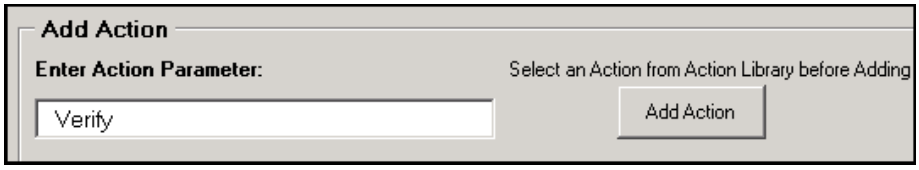


9. Select the rule's next action from the **Actions Library** sector.

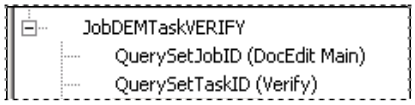
To Define a Rule (continued)

Step	Action

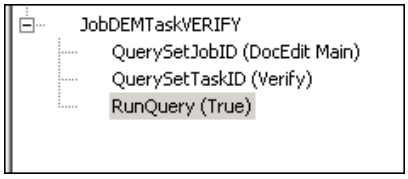
10. Enter the rule's parameter(s) in the **Enter Action Parameter** field and click on the Add Action button.



11. Confirm that the action and its parameter are now part of the rule:

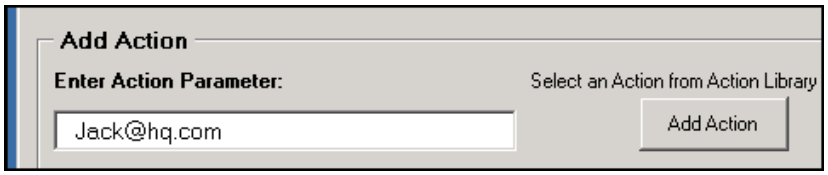


12. Where convenient and appropriate, select the **RunQquery** action from the library and enter *True* as its parameter. Click on the Add Action button and check to be sure the action is part of the rule:

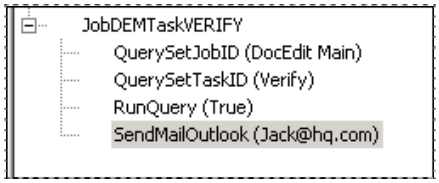


13. Select the all-important **SendMailOutlook** action (or **SendMailDONTS**.)

14. Enter the e-mail address of the communication's recipient (probably yourself at this early stage!) in the **Enter Action Parameter** field. ***Remember:*** Separate the addresses of multiple recipients with a comma (**SendMailOutlook)** or semicolon (**SendMailDONTS).**



15. Press the Add Action button and confirm that the action is part of the rule:



To Define a Rule (continued)

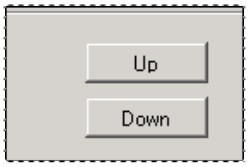
Step	Action

16. Review the rule and its actions as you prepare to give it a try.

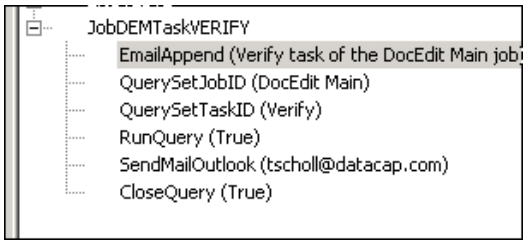
But wait just a moment! This rule lacks an **EmailAppend** action to give the communication a title. While this action is not required, it almost always increases the power of an ENU communication.

Inserting an action such as **EmailAppend**, which goes at the beginning of the rule, involves a little sleight of hand:

- Highlight the rule's name.
- Select the **EmailAppend** action from the **Actions Library**.
- Enter the text of the communication's title as the **Action Parameter** and click on the Add Actions button. ENU will add the action to the ***end*** of the rule.
- Use the *Batch Manager's* Up button to move the action to the top of the list.

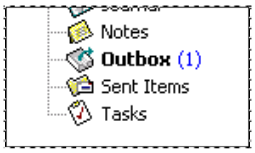


At this point, the rule should look like this:



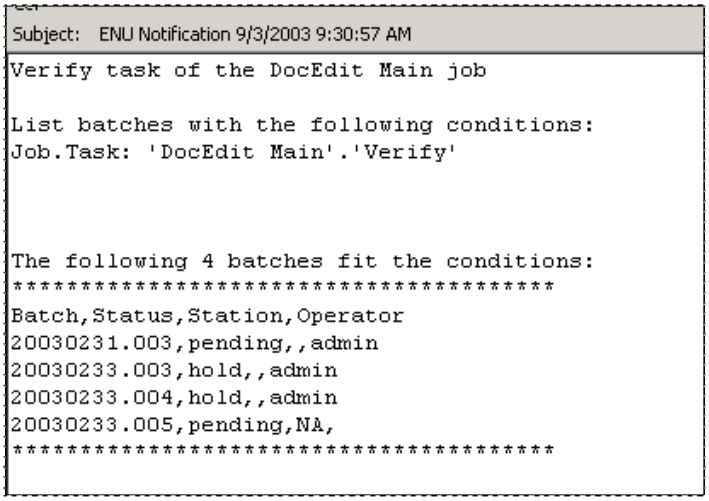
To run the rule, first select the rule, then press the Run button. (If you select the **Debug** option, ENU produces a more detailed log of the rule's activities.)

Open your *Email* window. If you're using *Outlook* – and the rule ran without difficulty – the ***Outbox*** will show one new entry:



Press *Outlook's* Send/Receive button to initiate mailing procedures:

☛ Almost immediately, a recipient's ***Inbox*** will include an e-mail with information similar to this:



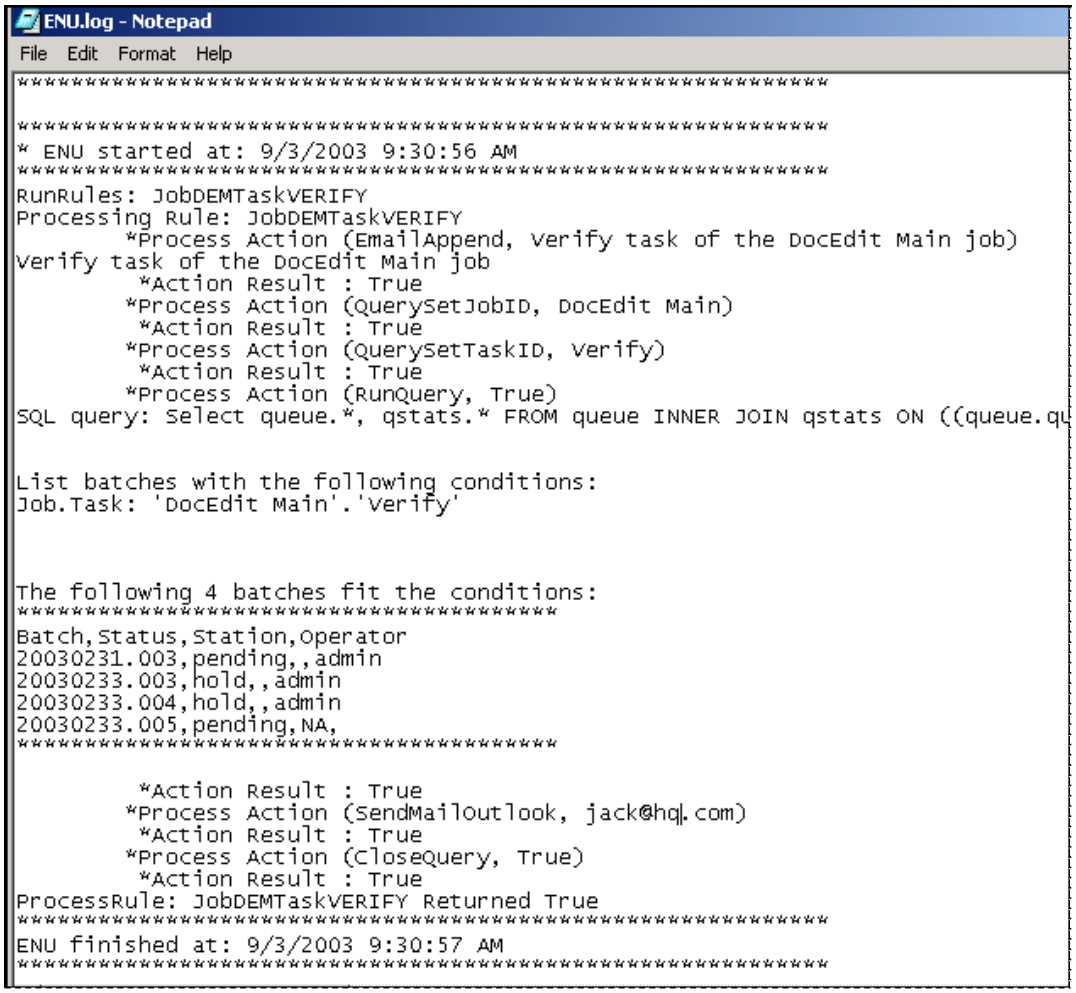
JobDEMTaskVERIFY Rule – *Communication*

✓ ***Important!*** If any action in a rule "fails" – if the action cannot function correctly and completely – the rule will stop and the ENU Log will record the problem. The next page explains the role and content of this helpful diagnostic tool.

Log Files

ENU automatically assesses the success of each action in a rule - and of the rule itself – and adds the results of its review to the end of a Log file the utility maintains for ***all*** rules and actions. The file is **ENU.log**; you'll find it in the directory that holds the other ENU files (Page 5).

As you can imagine, the log contains only positive information about a rule which runs without trouble. In the example we've been working with, the actions and the rule all return *True* conditions.



ENU Log – *JobDEMTaskVERIFY Rule*

However, the log's real value emerges when problems arise…when a parameter is missing or invalid, when an action occupies the wrong place in the rule's sequence, or when an action fails to perform.

The following page shows a portion of the ENU log which details a few of these problems.

```
**********************************************************************
* ENU started at: 9/3/2003 11:40:33 AM
**********************************************************************
RunRules: JobDEMTaskVERIFY
Processing Rule: JobDEMTaskVERIFY
        *Process Action (EmailAppend, Verify task of the DocEdit Main job)
Verify task of the DocEdit Main job
         *Action Result : True
        *Process Action (QuerySetJobID, DocEdit)
         *Action Result : True
        *Process Action (QuerySetTaskID, ExportTwo)
         *Action Result : True
        *Process Action (RunQuery, True)
SQL query: Select queue.*, qstats.* FROM queue INNER JOIN qstats ON ((queue


List batches with the following conditions:
Job.Task: 'DocEdit'.'ExportTwo'

No batches meet these conditions

**************************************
         *Action Result : False
ProcessRule: JobDEMTaskVERIFY Returned False
**********************************************************************
ENU finished at: 9/3/2003 11:40:33 AM
**********************************************************************
```

ENU Log - *JobDEMTaskVERIFY Rule (revised)*

In this intentionally abused version of the JobDEMTaskVERIFY rule, the Job ID parameter is *DocEdit* and the Task ID parameter is *ExportTwo*. Although the rule successfully searches for batches linked to this job and task, neither is a valid parameter. As a result, the log points out that "No batches meet these conditions"; that the Action Result is "False"; and that the rule itself "Returned False".

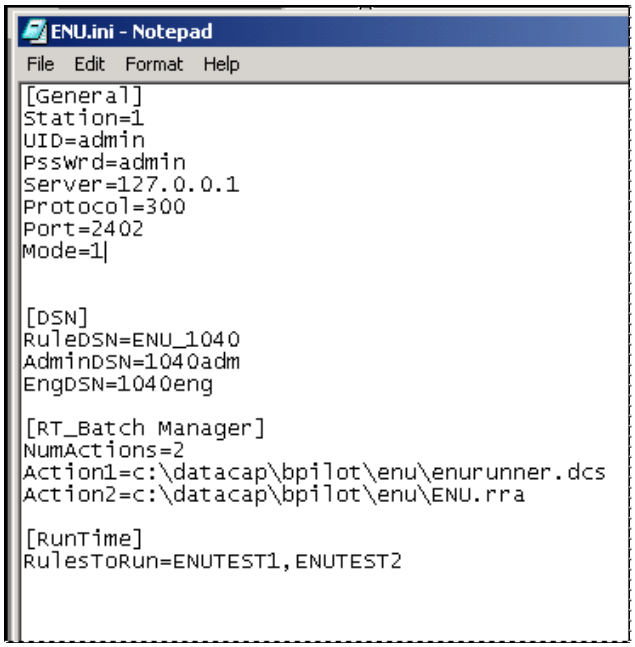
✓ Another indication that everything is not quite right shows up after you press the Run button, when you check the contents of your ***Outbox***. If the rule encounters an error of any sort, the rule will not instigate its e-mail procedures.

Scheduling ENU Communications

The ENU Settings file (**ENU.ini**) has a `[Runtime]` section with one entry:

```
RulesToRun = ENUTEST1, ENUTEST2
```

This entry designates which rules will run automatically, when the *Enhanced Notification Utility* is operating in a *scheduled* mode…when the INI file's **Mode** setting is "1". Eventually, when you have defined the rules you need, you'll try both modes. If you find that scheduling the generation of certain communications is a productive alternative, you can follow the steps in this section to set up the schedule and assign the rules it is to apply regularly.



ENU Settings File – *Scheduled Mode*

You can use the Windows ***Task Scheduler*** to determine when the *Enhanced Notification Utility* is to run.

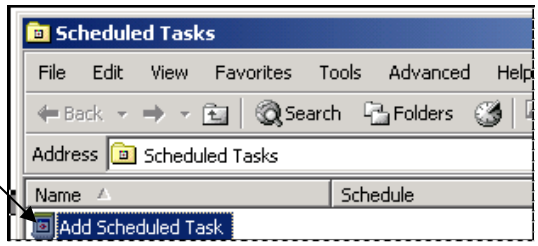
To define the schedule, take the steps outlined on the following page.

To Set up a Schedule

Step	Action

1. Click on your Windows **Start** button.

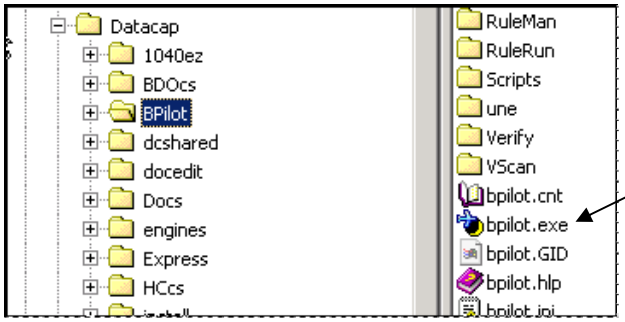
2. Select **Accessories** from the **Programs** options – and **Scheduled Tasks** from the **System Tools** options.



Double-click here

3. When the ***Scheduled Tasks*** window appears on your screen (above), double-click on *Add Schedule Task* listing in the window's Data Area: this opens the ***Scheduled Task Wizard.***

4. Click on the Next button of the *Welcome* screen, then on the Browse button to locate and open this file: **\Datacap\BPilot\bpilot.exe.**



Open this file.

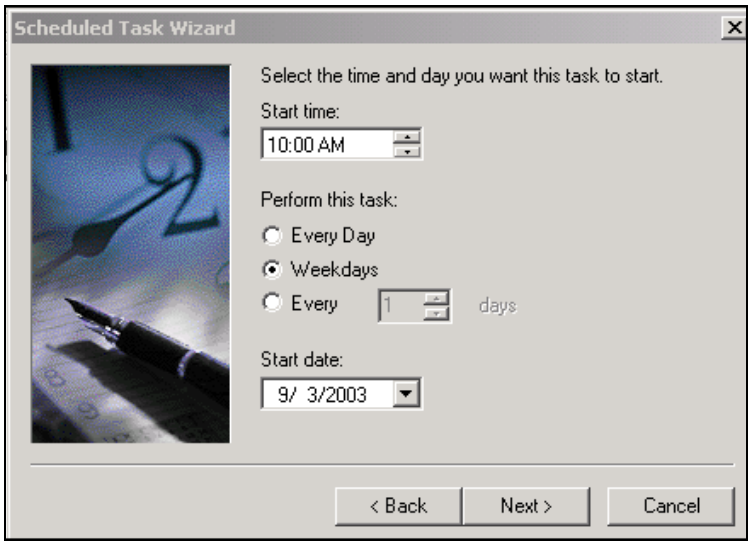
To Schedule ENU Communications (continued)

Step	Action

5. When the wizard returns, enter a name for this scheduling "task."

6. Select a **Frequency** option and press the Next button.

7. Add detailed scheduling parameters and click on the Next button.



To Schedule ENU Communications (continued)

Step	Action

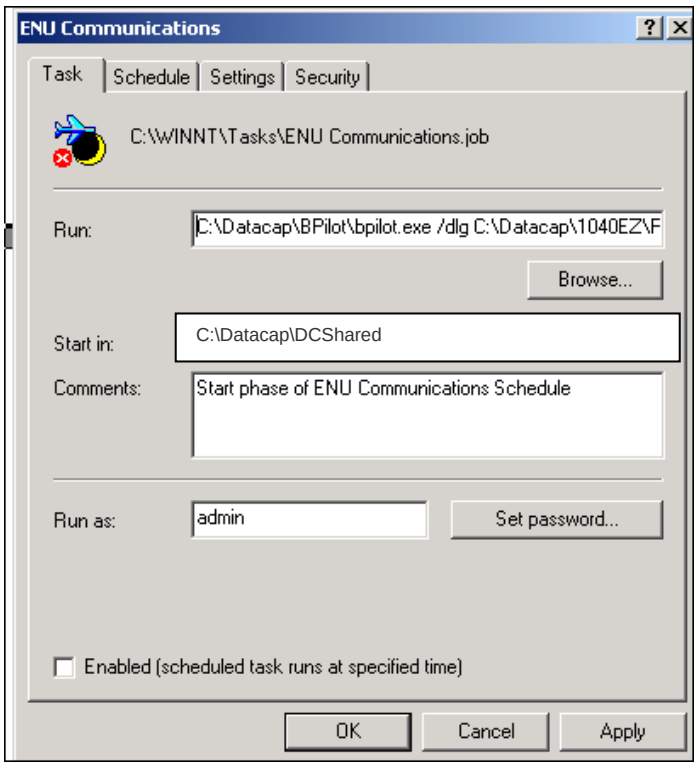
8. In the wizard's *Security* screen, complete the **User Name** and **Password** fields and press the Next button.

9. Review the criteria in the *Schedule Parameters* screen. Check the **Open Advanced Properties…** option and press the Finish button.

Select this option

To Schedule ENU Communications (continued)

Step	Action



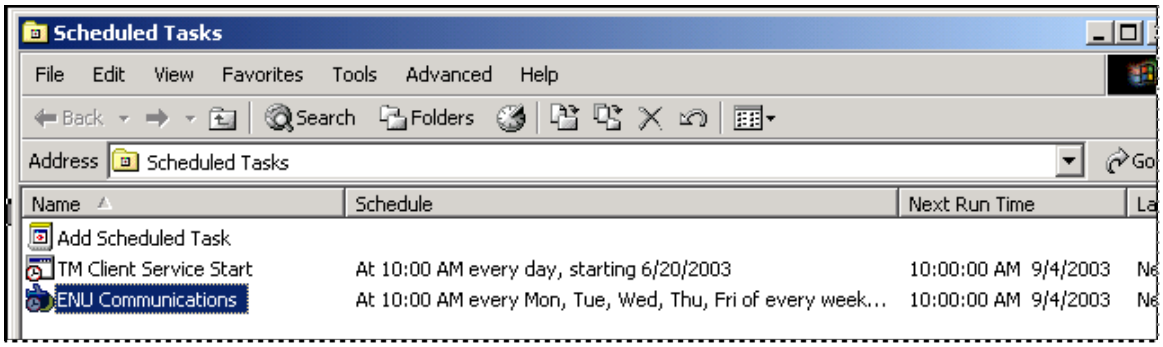
10. When the ***Advanced Properties*** dialog appears (above), enter this command line in the **Run** field (using your own drive letter, application ID and Directory names):

```
C:\Datacap\BPilot\bpilot.exe /dlg C:\Datacap
  \1040EZ\Process\enu\enu.dcf –rs enu.ini
```

11. In the **Start in:** field, enter:

```
C:\Datacap\DCShared
```

12. Click on the OK button. Confirm that the schedule is now part of the list of ***Scheduled Tasks*** (see the illustration on the next page.)



☛ At this point be sure to update the Settings file (ENU.ini) to reflect the requirements of the scheduling process:

List the rules that are to be scheduled in the `[Runtime]` section. For example:

```
RulesToRun=WaitingStatus,AbortStatus,Operator1
```

Change the `[General]` section's `Mode` value to "1".

✓ We ***strongly recommend*** that you test the new "scheduled task:"

- Highlight the task's title in the **Name** column of the ***Scheduled Tasks*** window (above).
- Choose **Run** from the window's **File** menu.

If a problem arises, review the latest entries in the Log file (**enu.log**) or look in the *Event Viewer* for possible errors.

To access the *Event Viewer*, click on your Windows Start button. From the **Programs** listings, select **Event Viewer** from the **Administrative Tools** options.