V. O. S. L.
Virtual Operator Script Language

Table of Contents

Overview

Introduction

Professional Programming Services is proud to introduce the latest in job automation & scheduling facility - “Virtual Operator”.

With Virtual Operator, it is now possible to run any series of programs or jobs (such as month end jobs or complex reports) automatically and without operator assistance or intervention.

How It Works

Virtual Operator is comprised of three modules:

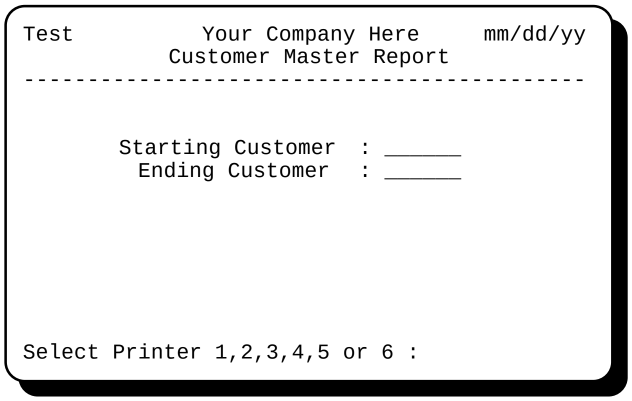
1. VOSM - Virtual Operator Scheduling Master. Used to schedule Virtual Operator scripts.
2. VORE - Virtual Operator Run Time Engine. Used to interpret and run Virtual Operator scripts.
3. VOSL - Virtual Operator Script Language. A user defined list of instructions or script, used by VORE to control an application program.

By attaching itself to a system level port, Virtual Operator can respond with information to any application program “as if” from a keyboard. Conversely, information sent or displayed to the screen by any application program can be monitored and evaluated by Virtual Operator.

With this communication link established, a user or system administrator can train Virtual Operator via the Virtual Operator Scripting Language (VOSL) to respond appropriately to an application's requirements. VOSL is a simple yet powerful and elegant scripting language used to automatically monitor and respond to input required by a specific application program.

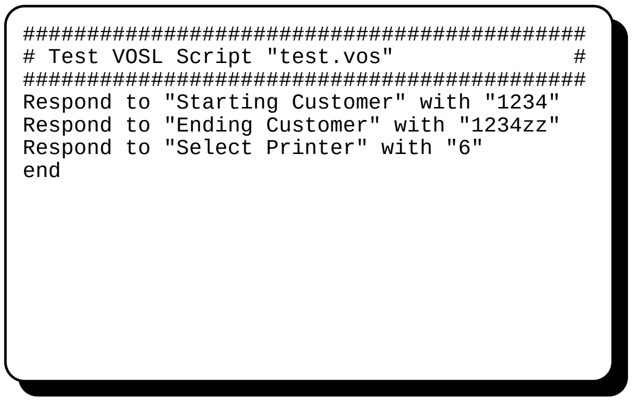
Virtual Operator Script Example

In this example (see figure 1), the application program expects the user to respond with three pieces of information: Starting Customer #, Ending Customer # and Printer Selection. The VOSL Script (see figure 2), will cause this application program to run a report of all customers whose codes begin with *1234*, to printer 6. VOSL can also trap for ERROR and TIMEOUT conditions.



```
Test          Your Company Here     mm/dd/yy
          Customer Master Report
-------------------------------------------


       Starting Customer  : ______
         Ending Customer  : ______









Select Printer 1,2,3,4,5 or 6 :
```

Figure 1



```
############################################
# Test VOSL Script "test.vos"              #
############################################
Respond to "Starting Customer" with "1234"
Respond to "Ending Customer" with "1234zz"
Respond to "Select Printer" with "6"
end
```

Figure 2

Conventions

The following conventions are used in this manual:

ALL CAPS	Used to denote a reserved word or System Variable.
Italics	Used to denote a string of characters sent to or received from an application program.
Bold Italics	Used to denote a directive's parameter.
[ ]	Used to denote an optional parameter.
ENTER	Used to denote the Enter or Return key.
F?	Used to denote a function key.

System Variables

Name	Value
ESC	The Escape Key (Hex 1B, Ascii 27).
CR	The Carriage Return or Enter Key (Hex 0D, Ascii 13).
LF	The Line Feed Key (Hex 0A, Ascii 10).
QUO	Double Quote Key (", Hex 22, Ascii 34).
F1 - F16	Functions 1 through 16 Keys.
TODAY	Today's Date in MM/DD/YY Format.
TODAY_MDY	Today's Date in MMDDYY Format.
TODAY_YMD	Today's Date in YYMMDD Format.
TOMORROW_MDY	Tomorrow's Date in MMDDYY Format.
TOMORROW_YMD	Tomorrow’s Date in YYMMDD Format.
LOGIN or NAME	User Unix Login Name (Specified in VOSM).
PASSWORD	User Unix Password (Specified in VOSM).
OPRC	IDOL Operator Code (Specified in VOSM).
OPRP	IDOL Password (Specified in VOSM).
NCR	NONE - (No Carriage Return) Used to Suppress ENTER with the RESPOND directive.

Directive: Facility for in-line script documentation.

PURPOSE This directive is used to denote the beginning of a comment.

SYNTAX **#[*text*]**

REMARKS All text following an unquoted # is ignored by Virtual Operator Run Time Engine until the end of that line.

EXAMPLES **#This is a comment**
In this example the entire line is ignored.

Respond to "#" with "Hello"   #Send Hello to the Application Program
In this example, the RESPOND directive is performed. All text following the second # is ignored.

Directive: Script reference point.

PURPOSE This directive is used as a script reference point allowing another directive to pass script control to a point occupied by this label.

SYNTAX ***:label_name***

REMARKS A label name must begin with a ":". The maximum size for a label is 15 characters. Valid characters are A-Z, a-z, 0-9, and the "_" (Underline Character). Labels are not case sensitive.

EXAMPLES

Goto :Block_1
Respond with "" # This line will be skipped by the above GOTO directive
:Block_1
In this example, script control will be passed to :Block_1 by the GOTO directive.

Wait for "Read Mail ?" Timeout=3 Fail=:Block_2
Respond with "N"
:Block_2
In this example, script control will be passed to :Block_2, if the WAIT directive FAILS to receive the string *Read Mail ?* from the application program.

SEE ALSO GOTO, RESPOND & WAIT Directives.

Directive: Transfer script control to a location specified by a label.

PURPOSE This directive is used to transfer script control to a location specified by a label.

EXAMPLE
Goto :Block_1
Respond with “” # This line will be skipped by the above GOTO directive
:Block_1
In this example, script control will be passed to :Block_1 by the GOTO directive.

SEE ALSO :LABEL Directive.

Directive: Sends data to an application program.

PURPOSE This directive is used to send data to an application program, "as if" from a keyboard. Optionally, this directive will wait for a trigger string (specified by the **TO** option), to be displayed by the application program to its "screen", prior to responding. Additionally, the data can be resent multiple times until a trigger string (specified by the **UNTIL** option) is satisfied.

SYNTAX **RESPOND [TO *string1*] WITH *string2* [UNTIL *string3*] [TIMEOUT=*value1*] [RETRY=*value2* [LABEL=*label1*]] [FAIL=*label2*]**

where:

string1 is one or more string literal(s) or variable(s) used by the directive as a trigger that must be satisfied before ***string2*** is sent to the application program.

string2 is one or more string literal(s) or variable(s) used by the directive to send data "as if" from a keyboard, to the application program.

string3 is one or more string literal(s) or variable(s) used by the directive as a trigger. The UNTIL option will cause the directive to resend the data in ***string2*** until ***string3*** is satisfied.

value1 is a integer literal (representing seconds) used by the directive to determine whether or not a TIMEOUT condition has occurred. A TIMEOUT condition will only occur when the TO or UNTIL options are used and not satisfied, and no data has been displayed to the "screen" by the application program for a time (in seconds) greater than that specified by ***value1***. The default value for ***value1*** is 30 if the RETRY option is omitted, or 1 if the RETRY option is specified.

value2 is a integer literal used by the directive to determine the maximum number of times (default 0) the current statement (or block, if the LABEL option is specified) is to be retried after a TIMEOUT condition has occurred. After ***value2*** retries, a subsequent TIMEOUT condition will cause a FAIL condition to occur.

label1 is a label referencing a location within the script. If the RETRY option is specified and a TIMEOUT condition occurs, script control will be passed to the location referenced by ***label1***. If the RETRY option is omitted the LABEL option is ignored.

label2 is a label referencing a location within the script. If a FAIL condition occurs, script control will be passed to the location referenced by ***label2***. If the FAIL option is omitted and a FAIL condition occurs, an ERROR condition will occur, causing the script to terminate with an error message result.

REMARKS

RESPOND will automatically append ENTER to the end of ***string2*** unless the system variable NCR (No Carriage Return) is specified as part of ***string2*** or if the last variable in ***string2*** is already an input terminator e.g. CR, F1, F2, F3, F4 etc.

EXAMPLES

Respond with "123zz" **# Send 123zz & <Return>**
Will send *123zz* ENTER to the application program.

Respond to "Change ?" with "3" F3 **# Send 3 with F3 key**
Will wait for the application program to print *Change ?* to its "screen" and then respond with *3* F3. If *Change ?* is not printed to the "screen" and the application program is "quiet" for more than 30 seconds, a FAIL condition will occur and the script will terminate with an error message.

:login **# Label login. Top of block.**
Respond with "" **# Send <CR> to wake up System**
Respond to "Login:" with NAME Retry=60 Label=:login Timeout=3
This script block will send ENTER to the application program. The script will then wait 3 seconds for the application program to respond or display the literal *Login:*. If *Login:* is displayed by the application program, the script will respond with the string value stored in the system variable NAME. Otherwise, the system will retry up to 60 times starting at label :login and resend ENTER until the word *Login:* is sent by the application program. If unsuccessful after 60 retries, a FAIL condition will occur and the script will terminate with an error message result.

Respond with "" Until "Login:" Retry=60 Timeout=3
Respond with NAME
This script block is functionally identical to the preceding example.

Respond to "You have mail" with "S" Fail=:Menu_Select
Respond with "Y"
:Menu_Select
This script block will wait for *You have mail* to be sent by the application program. If successful, the script will then Respond With an *S* ENTER (to save the mail) and then respond with a *Y* ENTER (confirming that the mail should be saved). If unsuccessful, script control will be passed to label **:Menu_Select**.

SEE ALSO

WAIT and :LABEL Directives.

Directive: Puts the script to sleep.

PURPOSE This directive is used to synchronize the script with the application program by waiting for a period of time, expressed in seconds.

SYNTAX **SLEEP *value1***

where:

value1 is a integer literal representing seconds.

REMARKS This statement is provided for compatibility purposes only and should be avoided. The TIMEOUT, RETRY & FAIL options in the RESPOND & WAIT directives should be used in its place.

EXAMPLE

```
Sleep 600                    # Sleep for 10 minutes
```

SEE ALSO RESPOND & WAIT Directives.

Directive: Waits for data to be sent by the application program.

PURPOSE This directive is used to synchronize the script with the application program by waiting for the application program to send a *trigger* string.

SYNTAX **WAIT [FOR] *string1* [TIMEOUT=*value1*] [RETRY=*value2* [LABEL=*label1*]] [FAIL=*label2*]**

where:

string1	is one or more string literal(s) or variable(s) used by the directive as a trigger that must be satisfied before continuing to the next script statement.
value1	is a integer literal (representing seconds) used by the directive to determine whether or not a TIMEOUT condition has occurred. A TIMEOUT condition will only occur when ***string1*** is not satisfied and no data has been displayed to the "screen" by the application program for a time (in seconds) greater than that specified by ***value1***. The default value for ***value1*** is 30 if the RETRY option is omitted or 1 if the RETRY option is specified.
value2	is a integer literal used by the directive to determine the maximum number of times (default 0) the current statement (or block, if the LABEL option is specified) is to be retried after a TIMEOUT condition has occurred. After ***value2*** retries, a subsequent TIMEOUT condition will cause a FAIL condition to occur.
label1	is a label referencing a location within the script. If the RETRY option is specified and a TIMEOUT condition occurs, script control will be passed to the location referenced by ***label1.*** If the RETRY option is omitted the LABEL option is ignored.
label2	is a label referencing a location within the script. If a FAIL condition occurs, script control will be passed to the location referenced by ***label2***. If the FAIL option is omitted and a FAIL condition occurs, an ERROR condition will occur, causing the script to terminate with an error message result.

EXAMPLES

```
Wait For "Login:"              # Wait for a login prompt
Respond with NAME              # Respond with Login Name
```

Will wait for the application program to display the literal *Login***:**, before passing script control to the proceeding **RESPOND** directive.

Wait for "Mail ?" Timeout=10 FAIL=:Menu_Select

Will wait for *Mail ?* to be sent by the application program. If successful, the script will continue with the next script statement. If unsuccessful, script control will be passed to label **:Menu_Select**.

SEE ALSO RESPOND and :LABEL Directives.

V. O. S. M.
Virtual Operator Scheduling Master

Nightly Batch Run Master Scheduler

Introduction

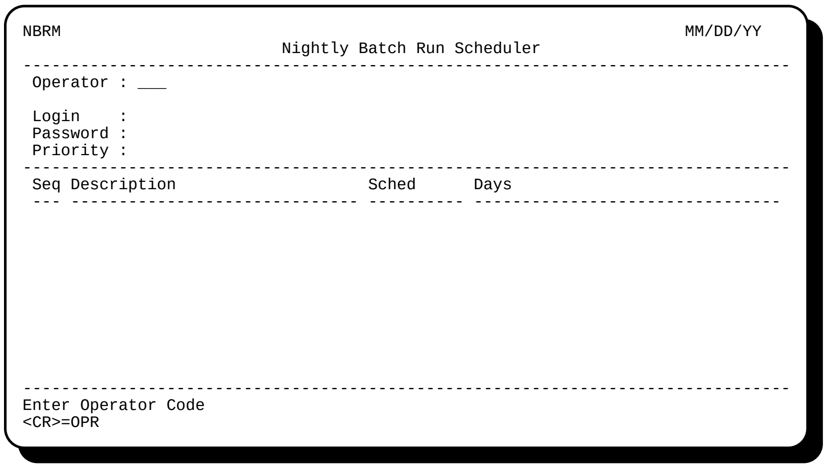
The Nightly Batch Run Master Scheduler allows a system administrator to:

A. Schedule a Virtual Operator Script.

B. Group scheduled scripts by user identity.

C. View the status or result messages of scripts that have already run.

D. Change or Delete a scheduled script.



```
NBRM                                                                 MM/DD/YY
                           Nightly Batch Run Scheduler
--------------------------------------------------------------------------------
 Operator : ___

 Login    :
 Password :
 Priority :
--------------------------------------------------------------------------------
 Seq Description                    Sched      Days
 --- ------------------------------ ---------- -------------------------------








--------------------------------------------------------------------------------
Enter Operator Code
<CR>=OPR
```

Figure 3

Description

The main screen (see figure 3) is divided into two sections. The upper section is used to associate one or more scripts with a specific user identity. This upper section contains four prompts:

Operator	Cimpro / IDOL Menu System Operator Code - A three character user code used to log in to Cimpro. The information entered here will be stored in the system variable OPRC and may be used within the scripts scheduled under this section. The variable OPRP containing the appropriate IDOL Operator Password will automatically be set.
Login	Unix Login Name - A name used to log in to Unix. The information entered here will be stored in the system variable LOGIN and may be used within the scripts scheduled under this section.
Password	Unix Password - A Password used to log in to Unix. The information entered here will be stored in the system variable PASSWORD and may be used within the scripts scheduled under this section.
Priority	A number ranging for 0 to 999 - Used to give priority to one user's set of schedules over another user's set of schedules. A user schedule containing a lower priority value will be executed first.

Nightly Batch Run Master Scheduler (Continued...)

The lower section contains information regarding what script will be run and when. This section contains nine prompts:

Seq	System Generated Sequence. Displays the order in which scripts will be run within the current user identity.
Description	Used as documentation in describing the scheduled script.
Sched Option	Used in conjunction with "Day List". Allows a script to be scheduled as a: 1. One time only event 2. Repeating event based on one or more days of the week (e.g. Mondays, Wednesdays & Fridays). 3. Repeating event based on one or more days of the month (e.g. the 1st & the 15th of every Month).
Day List	Used in conjunction with "Sched Option" to specify dates in which the script will run.
Starting	Used to specify the starting date for this event. A scheduled script will begin to run on or after this specified date.
Program	Used to specify the Virtual Operator Script to be run.
Dependencies	Used to specify a dependency between the current script and other scripts. The current script (specified in "Program") will only run if the scripts (specified in "Dependencies") were successful.
Last Run	System generated display field containing the date in which this script was last run.
Result	System generated display field containing information as to the success or failure of a scheduled script.