

FORTRA



Robot Console OPAL Reference Guide

Copyright Terms and Conditions

Copyright © Fortra, LLC and its group of companies. All trademarks and registered trademarks are the property of their respective owners.

The content in this document is protected by the Copyright Laws of the United States of America and other countries worldwide. The unauthorized use and/or duplication of this material without express and written permission from Fortra is strictly prohibited. Excerpts and links may be used, provided that full and clear credit is given to Fortra with appropriate and specific direction to the original content.

202301100343

Table of Contents

| | |
|--|-----------|
| OPAL Language | 5 |
| The OPerator Assistance Language (OPAL) | 5 |
| OPAL Code in Message Sets: Before and After | 5 |
| OPAL Statements | 6 |
| OPAL Fields | 6 |
| Logic Operands | 7 |
| Using OPAL Variables | 20 |
| Using OPAL Message Data Variables | 45 |
| Comparisons | 58 |
| Operations | 60 |
| Operation Values | 90 |
| OPAL Source Code | 93 |
| Examples | 96 |
| OPAL Code Examples | 96 |
| OPAL Example 1: Tape Device Error Can Be Recovered | 97 |
| OPAL Example 2: Message Queue Deleted or Damaged | 98 |
| OPAL Example 3: Workstation Varied Off Due to Wrong Password Entry | 98 |
| OPAL Example 4: Device Problem Messages Redirected to QSECOFR | 99 |
| OPAL Example 5: Receive Save Files and Restore Objects | 99 |
| OPAL Example 6: TCP Job Ended | 100 |
| OPAL Example 7: User Not Authorized to Change System Value | 101 |

| | |
|--|------------|
| OPAL Example 8: Cannot Allocate Library | 101 |
| OPAL Example 9: Line Failed—Probable Local Hardware Problem | 102 |
| OPAL Example 10: Produce a Dump | 103 |
| OPAL Example 11: Answer Message and Remove from Message Center | 103 |
| OPAL Example 12: Change Informational Message to Response Required | 104 |
| OPAL Example 13: Use a Notification List | 104 |
| OPAL Example 14: Use RBTBCHUPD to Run a Robot Schedule Job | 104 |
| OPAL Example 15: Check OPAL Table for Variable Value | 105 |
| OPAL Example 16: Send a System Alert and Break Message | 105 |
| OPAL Example 17: Change Message to Response Required | 106 |
| OPAL Example 18: Answer Message Depending on Device Name | 106 |
| OPAL Example 19: Send Message or Reset Device | 107 |
| OPAL Example 20: Suppress a Message or Send a Message | 107 |
| OPAL Example 21: End a Job | 107 |
| OPAL Example 22: Answer a Repeated Message | 108 |
| OPAL Example 23: Redirect Messages to an Active User | 108 |
| OPAL Example 24: Redirect Messages to HOSTCENTER | 108 |
| Quick Reference Guide | 109 |
| Robot Console Quick Reference Guide | 109 |

OPAL Language

The OPerator Assistance Language (OPAL)

Robot Console provides a customized language for message processing. The language is called OPAL, the OPerator Assistance Language. OPAL lets you define specific processing for the messages handled by a message set.

The OPAL language is easy to learn. It's a fixed format language, like RPG, but its syntax is like CL. To code OPAL statements, you just fill in fields on a panel. Help is available for each field, and Robot Console checks what you enter. Error messages generated by OPAL statements are placed in the OPAL joblog.

If you don't find the information you need in this reference guide, go to the [Robot product page](#) on our website. There you can find additional information in our support topics or access contact information needed to connect with a Robot Technical Consultant.

OPAL Code in Message Sets: Before and After

A message set can contain two sections of OPAL code: OPAL code performed only before a message reply is received and OPAL code performed only after a reply is received. You can enter OPAL code in a message set on either the IBM i or the Robot Console Explorer window (GUI). To enter OPAL code:

- IBM i: Display the OPerator Assistance Language panel (option 4 from the Maintain Message Sets panel) to enter before-reply OPAL code. From that panel, press function key 15 to enter after-reply OPAL code.
- GUI: Under Message Sets in the Tree view, click **All Message Sets**. Then, right-click a message set in the List view, and select either **New** (to add a new set) or **Properties** (to edit the existing set). Then, click the **OPAL** tab to enter before-reply code, or click the **Reply OPAL** tab to enter after-reply code.

If a reply has not been received when the before-reply OPAL code completes, Robot Console does its automatic notification procedures to try to get a reply. As described for the Message Centers tab of the System Defaults window of the *Robot Console User Guide*, these procedures can include sending pager messages and redirecting the message to another message center.

If the message set in use contains reply OPAL code, the message set waits as long as necessary to get a reply. It then executes the reply OPAL code. If the message set does not

contain reply OPAL code, the message set completes after the auto-notification procedures are done. Message processing can then continue with the next message set if any.

OPAL Statements

OPAL code consists of a sequence of OPAL statements. The OPAL statements are performed in order unless an IF, WHILE, or GOTO statement changes the processing order.

An OPAL statement is entered on a single line. (OPAL does not use continuation lines.) Each line has several fields as shown in the following example:

| RBC275 | | Operator Assistance Language | | 17:34:53 | |
|----------------------------------|----------|------------------------------|-------------------------------------|--------------------|--|
| Message Group Name . . . : *NONE | | | LIL | | |
| Message Set Name . . . : TWTEST | | | Status: Released | | |
| Monitor for Message ID : CPF0907 | | | Testing OPAL code | | |
| Start at Sequence . . . : ____ | | | | | |
| Logic | | | | | |
| Operand | Variable | Operation | Operation Values | Seq | |
| IF | WORKDAY | EQ | Y | 10 | |
| IF | SYSTIME | GT | 170000 | 20 | |
| OR | SYSTIME | LT | 070000 | 30 | |
| | | * | If workday and after working hours | 40 | |
| THEN | | DELAY | 1800 | 50 | |
| | | * | If working hours, delay 10 mins | 60 | |
| ELSE | | | | 70 | |
| | | DELAY | 600 | 80 | |
| END | | | | 90 | |
| | | * | If working day, vary on workstation | 100 | |
| | | EXECUTE | VRYCFG CFGOBJ(VAR3) CFGTYPE(*DEV... | 110 | |
| | | * | | 120 | |
| | | | | More... | |
| F3=Exit | | F4=Prompt | | F7=ROBOT Variables | |
| F12=Previous | | F15=Reply OPAL | | F24=More Keys | |

The contents of each field is indicated by its heading. On the IBM i, the last column (Seq) is not part of the OPAL statement; it contains the sequence number of the statement.

These four fields constitute the OPAL statement. Each OPAL example in this reference guide uses a table similar to the one below to show you the fields used by the statement:

| Logic Operand | Variable | Operation | Operation Values |
|------------------|----------|-----------|---------------------|
| | | | |

OPAL Fields

The following sections describe what you can enter in the OPAL statement fields. The fields are described in the same order that they occur in an OPAL statement: Logic Operands, Variables, Operations, and Operation Values.

The first field in an OPAL statement is the Logic Operand field. Logic operands are used to define logic control for the OPAL code, that is, which OPAL statement is performed next.

Logic Operands

The first field in an OPAL statement is the Logic Operand field. Logic operands are used to define logic control for the OPAL code; that is, which OPAL statement is performed next.

Three logic control structures are available in OPAL: IF, WHILE, and GOTO. The IF and WHILE structures specify conditions that must be met before a set of operations is performed. A GOTO operation changes the next statement processed to the specified TAG statement.

Logic Control

You don't need logic control in your OPAL code if all operations in the code are to be performed for every message processed by the OPAL code. For example, the following OPAL code consists of just two unconditional operation statements. The first statement changes the message so it requires a response. The second statement requires that the person on whose message center the message appears respond to the message.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| | | RESPOND | |
| | | OPERATOR | |

Conditional Operations

To make operations conditional, that is, conditions must be met before the operations are performed, you create IF and WHILE structures in your OPAL code. An IF or WHILE structure specifies the conditions that must be met and the operations that are performed if the conditions are met. IF and WHILE structures use the logic operands described on the following pages.

The following is an example of an IF structure. The operation in the IF structure is performed only if the conditions are met. In this case, DAVEJ is paged only if the system time is greater than 5 pm or less than 8 am or today is not a workday.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | SYSTIME | GT | 170000 |
| OR | SYSTIME | LT | 080000 |
| OR | WORKDAY | EQ | NO |
| THEN | | PAGE2WAY | DAVEJ |
| END | | | |

Logic Operand: IF

IF—Perform If Conditions are Met

Use an IF statement to test for one or more conditions. If the conditions are true, the operations following the condition list are performed. If the conditions are not true, the program continues at the statement following the operation list.

If the operations should be performed just once if the conditions are true, use an IF statement. Use a WHILE statement if the operations should be performed more than once.

Syntax

| Logic Operand | Variable | Operation | Operation Values |
|---------------|------------|------------|------------------|
| IF | (required) | (required) | (required) |

Examples

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | REPEAT | LE | 7 |
| IF | ACTUSR | EQ | DAVEJ |
| IF | DATA | CT | SALUPD |

Multiple Conditions

You can add more conditions to the IF statement by using AND and OR statements after the IF.

An optional THEN statement can mark the end of the condition list and the beginning of the operations to be performed if the conditions are true.

For example, the following statements send a message to pager OPERATOR if DAVEJ is not an active user and the time is either after 5 p.m. or the day is not a workday:

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | ACTUSR | NE | DAVEJ |
| AND | SYSTIME | GT | 170000 |
| OR | ACTUSR | NE | DAVEJ |
| AND | WORKDAY | EQ | NO |
| THEN | | PAGE2WAY | OPERATOR |
| END | | | |

Nested IF Statements

IF statements can be nested. That is, an IF statement can be enclosed in another IF statement. For example, the above example code could be rewritten as a nested IF structure as follows:

Nested IF statements

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | ACTUSR | EQ | DAVEJ |
| IF | SYSTIME | GT | 170000 |
| OR | WORKDAY | EQ | NO |
| THEN | | PAGE2WAY | OPERATOR |
| END | | | |
| END | | | |

The highlighted IF statement is nested within the other IF statement.

END Statements Required

For every IF statement, there must be a corresponding END statement. However, END statements can be omitted before ELSE statements and at the end of the program. Consider this program structure:

| | | |
|---|------|----------------|
| 1 | IF | condition list |
| 2 | THEN | operation list |
| 3 | END | |
| 4 | ELSE | |
| 5 | IF | condition list |
| 6 | THEN | operation list |
| 7 | END | |
| 8 | END | |
| 9 | | operation list |

- The END in line 3 is not necessary because END is not required before ELSE.
- The END in lines 7 is required because this is not the end of the program.
- The END in line 8 is required to end the nested IF structure.
- The 'operation list' in lines 2, 6, and 9 are not part of the nested IF structure.

Logic Operand: WHILE

WHILE—Do WHILE Conditions are Met

The WHILE statement in OPAL is the same as the Do While (DOW) operation code in RPG/400. Use the WHILE statement for operations that are to be repeated until the conditions are not true.

A WHILE structure tests one or more conditions. If the conditions are true, the operations following the condition list are performed, then the conditions are tested again. When a test finds that the conditions are not true, the program continues at the statement following the WHILE operation list.

Syntax

| Logic Operand | Variable | Operation | Operation Values |
|---------------|------------|------------|------------------|
| WHILE | (required) | (required) | (required) |

Examples

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| WHILE | USRNBR1 | LE | 7 |
| WHILE | USRFLG1 | EQ | Y |

Exiting the WHILE Loop

OPAL can exit a WHILE loop only when one of these events happens:

- The WHILE condition list becomes false after it was initially true.
- A GOTO operation is performed that jumps out of the loop.
- A RPYWITHIN operation is in progress and a reply is received.

Note: A RPYWITHIN operation must be in progress when the reply is received; otherwise, the reply does not cause OPAL to exit the loop.

Change WHILE Condition in Loop

The condition list on the WHILE statement should be one that, if true, will change to false so processing can leave the loop. Therefore, one of the loop conditions should generally be a user variable that you explicitly set before the loop and change within the loop.

For example, the following code tries three times to vary on an inactive line:

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|--------------------------|
| | USRNBR1 | CHGTO | 3 |
| WHILE | USRNBR1 | GT | 0 |
| AND | ACTLIN | NE | LINE |
| | | EXECUTE | VRYCFG (CFGOBJ (LINE)... |
| | USRNBR1 | SUB | 1 |
| END | | | |

If OPAL Can't Exit the Loop

If the OPAL processing cannot exit the loop, it cannot stop processing on its own. If this happens, you'll need to end the OPAL job. To do so, display the Robot Console Control Menu

and select the option to display all active message sets. Find the message group and message set name and end the message set (it doesn't end the job that sent the message).

To avoid this problem, carefully test your message sets using test data and the trace function as described in the Message Sets section of the *Robot Console User Guide*.

Multiple Conditions

Like the IF statement, you can add more conditions to the WHILE statement using AND and OR statements after the WHILE.

An optional THEN statement can mark the end of the condition list and the beginning of the operations to be performed if the conditions are true.

Nested WHILE Statements

You can also nest WHILE statements, that is, a WHILE loop can be enclosed in another WHILE loop.

END Statements Required

Each WHILE statement must have a corresponding END statement. END statements are implied at the end of the program

Nested WHILE Loop Example

The following example illustrates nested WHILE loops. Assume that the message processed by the program contains a line name. While DAVEJ is logged on, the program tries three times to vary off and then vary on the line.

Nested WHILE loops

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|--------------------------|
| WHILE | ACTUSR | EQ | DAVEJ |
| THEN | USRNBR1 | CHGTO | 0 |
| WHILE | ACTLIN | NE | LINE |
| AND | USRNBR1 | LT | 3 |
| THEN | | EXECUTE | VRYCFG (CFGOBJ (LINE)... |
| | | EXECUTE | VRYCFG (CFGOBJ (LINE)... |

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|--------------------------|
| | USRNBR1 | ADD | 1 |
| END | | | |
| IF | USRNBR1 | GE | 3 |
| THEN | | SENDMC | Press F4 to see command. |
| | | DELAY | 300 |
| END | | | |
| | | SENDMC | Press F4 to see command. |
| END | | | |

The highlighted WHILE and IF statements are nested within the outer WHILE statement to create a WHILE loop.

Logic Operand: AND

AND—Add a Condition to a Condition Set

Use an AND statement to add another condition to a set of conditions. (In contrast, an OR statement starts a new condition set.) All conditions in a set must be true for a true result.

You can also use the AND statement to add a condition to an IF or WHILE condition list. The AND connects the condition to the preceding condition in the list. Both conditions must be true for a true result; if either condition is false, the result is false.

Syntax

| Logic Operand | Variable | Operation | Operation Values |
|---------------|------------|------------|------------------|
| AND | (required) | (required) | (required) |

Examples

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| AND | ACTLIN | NE | LINE |

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| AND | USRNBR1 | LE | 7 |
| AND | DATA | CT | EMPMAS |

Any Number of Conditions in a Set

A condition set can contain any number of conditions. For example, the following IF statement has three conditions (highlighted below) that create a set of conditions that must all be true for the operation to be performed:

One set, 3 conditions in set

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | ACTLIN | NE | LINE |
| AND | ACTUSR | NE | GEORGE |
| AND | WORKDAY | EQ | NO |
| THEN | | PAGE2WAY | GEORGE |
| END | | | |

Logic Operand: OR

OR—Start New Condition Set

Use an OR statement to start a new set of conditions. You can also use the OR statement to start a new set of conditions in an IF or WHILE condition list. **Note:** To add a condition to an existing set of conditions instead, use the AND statement.

Syntax

| Logic Operand | Variable | Operation | Operation Values |
|---------------|------------|------------|------------------|
| OR | (required) | (required) | (required) |

Examples

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| OR | ACTDEV | NE | DEVICE |
| OR | USRFLG1 | EQ | X |
| OR | TEXT | CT | Ledger |

OR Starts a New Condition Set

Each OR statement in an IF or WHILE condition list starts a new set of conditions. For example, the OR statement in the following IF condition list starts a second condition set. Each highlighted set contains one condition; if either condition is true, the result is true.

2 sets, 1 condition in each set

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | ACTUSR | NE | DAVEJ |
| OR | ACTUSR | NE | GEORGE |
| THEN | | PAGE2WAY | DAVEJ |
| END | | | |

Each Condition Set Tested Separately

Each condition set is tested separately. For example, in the following IF condition list, one of the conditions is the same in both of the highlighted condition sets. But, for the condition set to be true, the other condition in the set must also be true.

2 sets, 2 conditions in each set

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | ACTUSR | NE | DAVEJ |
| AND | SYSTIME | GT | 170000 |
| OR | ACTUSR | NE | DAVEJ |
| AND | SYSTIME | LT | 090000 |
| THEN | | PAGE2WAY | DAVEJ |
| END | | | |

Rewrite as Nested IF

When the same condition is duplicated in all condition sets, it can be pulled out and put in its own IF. Thus, the above code could be rewritten as a nested IF as follows:

Nested IF statements

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | ACTUSR | NE | DAVEJ |
| IF | SYSTIME | GT | 170000 |
| OR | SYSTIME | LT | 090000 |
| THEN | | PAGE2WAY | DAVEJ |
| END | | | |
| END | | | |

The highlighted IF statement is nested within the other IF statement.

Logic Operand: THEN

THEN—Start Operation List

The THEN statement is always optional. Use it to mark the beginning of an operation list after an IF, WHILE, or ELSE structure. Only one THEN is allowed for each IF, WHILE, or ELSE.

Syntax

| Logic Operand | Variable | Operation | Operation Values |
|---------------|------------|------------|------------------|
| THEN | (optional) | (required) | (required) |

Examples

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|----------------------------|
| THEN | | PAGE2WAY | DAVEJ |
| THEN | | EXECUTE | VRYCFG CFGOBJ (LINE)... |
| THEN | | DEFAULT | |

Use in IF Structure

THEN can be used to mark the beginning of each operation list in an IF-ELSE structure as follows:

| | |
|------|----------------|
| IF | condition list |
| THEN | operation list |
| ELSE | |
| THEN | operation list |
| END | |

Use in WHILE Structure

THEN can mark the beginning of the operation list in a WHILE structure as follows:

| | |
|-------|----------------|
| WHILE | condition list |
| THEN | operation list |
| END | |

Logic Operand: ELSE

ELSE—If Conditions Are Not Met

Use the ELSE condition to extend an IF structure so it includes processing to be done only if the preceding IF condition lists are not true.

Syntax

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| ELSE | (blank) | (blank) | (blank) |

Example

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| ELSE | | | |

Every ELSE Needs an END

Each ELSE statement must be followed by a corresponding END statement. However, the END can be omitted before another ELSE or at the end of the program.

The simplest IF-THEN-ELSE structure is as follows:

| | | | |
|------|----------------|----------------|--|
| IF | condition list | | |
| THEN | | operation list | Operations performed if the condition list is true. |
| END | | | |
| ELSE | | | |
| THEN | | operation list | Operations performed if the condition list is false. |
| END | | | |

Extends IF for Additional Conditions

Additional ELSE clauses can be added to process other conditions. Only one operation list in the IF structure is executed. For example:

| | | | |
|-------|------------------|----------------|--|
| IF | condition list 1 | | |
| THEN | | operation list | Done if condition list 1 is true. |
| END | | | |
| ELSE | | | |
| IF | condition list 2 | | |
| THEN | | operation list | Done if condition list 1 is false, but condition list 2 is true. |
| ELSE | | | |
| END | | | |
| END | | | |
| WHILE | condition list 3 | | |

| | | | |
|------|--|----------------|--|
| THEN | | operation list | Done if condition lists 1 and 2 are false, while condition list 3 is true. |
| END | | | |
| END | | | |
| THEN | | operation list | Done if condition lists 1 and 2 are false, and the first test of condition list 3 is also false. |
| END | | | |

Logic Operand: END

END—Ends the Operation List

Every IF, WHILE, and ELSE statement must be followed by a corresponding END statement. The END statement marks the end of the processing for the IF, WHILE, or ELSE.

However, an END statement is optional if:

- It is immediately before an ELSE.
- It is at the end of the program.

Syntax

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| END | blank | blank | blank |

Example

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| END | | | |

When END is Optional

The following outline shows where END statements are optional:

| | | |
|---|----|----------------|
| 1 | IF | condition list |
|---|----|----------------|

| | | |
|----|------|----------------|
| 2 | THEN | operation list |
| 3 | END | |
| 4 | ELSE | |
| 5 | IF | condition list |
| 6 | THEN | operation list |
| 7 | END | |
| 8 | ELSE | |
| 9 | THEN | operation list |
| 10 | END | |
| 11 | END | |

- The END in lines 3 and 7 aren't necessary because END is not required before ELSE.
- The END in lines 10 and 11 are optional because they're at the end of the program.

So, if you remove the optional END operands, the remaining code is as follows:

| | | |
|------|----------------|----------------|
| IF | condition list | |
| THEN | | operation list |
| ELSE | | |
| IF | condition list | |
| THEN | | operation list |
| ELSE | | |
| THEN | | operation list |

Using OPAL Variables

This section explains what should be entered in the Variable field and lists the various OPAL variables that can be used.

Variable Field

In an OPAL statement, you use the second field, that is, the Variable field, for either of the following:

- To specify the variable to be tested by an IF, WHILE, AND, or OR statement.
- To specify the variable to be changed by a CHGTO, ADD, or SUB operation.

For all other statements, the field is left blank. The CHGTO, ADD, and SUB operations are described later under [Operations](#).

Conditions

To specify a condition on an IF, WHILE, AND, or OR statement, you specify three values:

- In the **Variable field**, the OPAL variable with the value to be tested.
- In the **Operation field**, a comparison operator.
- In the **Operation Values field**, an operation value that's either the actual value or an OPAL variable.

When the condition is tested, the values of the Variable and Operation Values fields are compared. If the two values have the relationship specified by the comparison operator, the condition is true. Otherwise, the condition is false.

For example, consider the following IF condition:

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | SYSTIME | GT | 170000 |

SYSTIME is an OPAL variable containing the current system time; the comparison operator is GT; and the operation value is 170000. When the condition is tested, the current system time is compared with the value 170000. If the time is greater than 170000, the condition is true; otherwise, it is false.

OPAL Variables

OPAL variables contain values that are obtained from the message being processed. In most cases, the variable receives its value when Robot Console begins processing the message. You can use an OPAL variable in the Variable field, in the Operation Values field, or in a command parameter in the Operation Value field.

For example, all of the following statements use the OPAL variable LINE:

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | LINE | EQ | DALLAS |
| THEN | USRFLD1 | CHGTO | LINE |
| | EXECUTE | VRFCFG | CFGOBJ(LINE)... |

Check What's Active

You can use the following OPAL variables to check if a system component or user is active. Compare the OPAL variable to a name; the name can be taken from the message data.

- ACTUSR
- ACTLIN
- ACTCTL
- ACTDEV

These variables are described in the following sections.

Note: Unlike the other OPAL variables, these four variables do not represent values and, therefore, cannot be used in the Operation Values field.

ACTUSR—Is This User Active?

The **ACTUSR** variable checks whether a user is active. Robot Console considers a user active if he has signed on and executed the **RBCSTRQ** command. (**Note:** The **RBCSTRQ** command could be executed automatically as part of the initial program for the user profile. See the System Setup section of the *Robot Console User Guide* for more information.) If the user is not active, you could redirect the message to another user or, if you have Robot Alert installed, you could send an email message to the user.

Operation: The only valid comparisons are EQ and NE: EQ is true if the user is signed on and has executed the **RBCSTRQ** command; NE is true if the user is not signed on or is signed on but has not executed the **RBCSTRQ** command.

Value: Name of the user profile to be checked.

Example: The following condition is true if user profile DAVEJ is not signed on.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | ACTUSR | NE | DAVEJ |

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| THEN | | PAGE2WAY | OPERATOR |
| END | | | |

ACTLIN—Is This Line Active?

The **ACTLIN** variable checks whether the status of the communication line is active. The communication line must be active; if the line status is VARIED ON or PENDING, it is not considered to be active.

If the line is not active, you could execute a VRYCFG command to vary off the line and all downline controllers and devices, and then execute another VRYCFG command to vary on the line and all downline controllers and devices. This should clear all error conditions for the line.

Operation: The only valid comparisons are EQ and NE: EQ is true if the line is active; NE is true if the line is not active.

Value: Name of the communication line to be checked.

Example: The following condition is true if the line is not active. The line name is the one in the message.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | ACTLIN | NE | LINE |

ACTCTL—Is This Controller Active?

The **ACTCTL** variable checks whether the status of a controller is active. A controller is a processor that controls one or more I/O devices, such as display stations or tape units. The controller must be active; if the controller status is VARIED ON or PENDING, it is not considered to be active.

If the controller is not active, you could execute a VRYCFG command to vary off the controller and all downline devices, and then execute another VRYCFG command to vary on the controller and all downline devices. This should clear all error conditions for the controller.

Operation: The only valid comparisons are EQ and NE: EQ is true if the controller is active; NE is true if the controller is not active.

Value: Name of the controller to be checked.

Example: The following condition is true if the controller is not active. The controller name is the one in the message.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|--------------------------|
| IF | ACTCTL | NE | CONTROLLER |
| THEN | | EXECUTE | VRYCFG CFGOBJ (CON... |
| END | | | |

ACTDEV—Is This Device Active?

The **ACTDEV** variable checks whether the status of a device is active. An IBM i device can be a display station, printer, diskette unit, tape unit, or remote system. The device must be active; if the device is varied on or pending, it is not considered to be active.

If the device is not active, you could vary off the device, vary on the device, wait a few minutes, and then check again to see whether the device is active. You could code a WHILE loop to repeat this procedure a few times, if needed.

Operation: The only valid comparisons are EQ and NE: EQ is true if the device is active; NE is true if the device is not active.

Value: Name of the device to be checked.

Example: The following condition is true if the device is not active. The device name is the one in the message.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|---------------------------|
| IF | ACTDEV | NE | DEVICE |
| THEN | | EXECUTE | VRYCFG CFGOBJ (DEVI... |
| END | | | |

Get the System Time and Date

Use the following OPAL variables to reference the current date and time on the system. These variables contain the date and time that the OPAL statement is executed.

- SYSTIME
- SYSDATE

These variables are described in the following sections.

SYSTIME—Current System Time

The **SYSTIME** variable gets the current time from the system clock when the statement is processed. Use SYSTIME if the message processing should differ depending on the time of day.

Value: A six-digit number representing the time as hours, minutes, and seconds on a 24-hour clock. For example, 123000 is exactly one half hour after noon.

Example: The first condition is true if the current time is after 5:00 p.m., but before midnight. The second condition is true if the current time is after midnight, but before 8:00 a.m.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | SYSTIME | GT | 170000 |
| OR | SYSTIME | LT | 080000 |
| THEN | | PAGE2WAY | OPERATOR |
| END | | | |

SYSDATE—Current System Date

The **SYSDATE** variable gets the current system date when the statement is processed.

Value: A six-digit number representing the date in the system format. For example, if the system format is year, month, day (yymmdd), then 150210 is February 10, 2015. **Note:** If you compare the SYSDATE variable with a date in a ROBOT reserved command variable, make sure that the date in the ROBOT variable is in the year, month, day (yymmdd) format.

Example: The following condition is true if the current date is before February 1, 2015, and the system date format is yymmdd.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | SYSDATE | LT | 150201 |

Use the Robot Schedule Calendar

If you have Robot Schedule installed on your system, you can reference the characteristics of a date as defined by a Robot Schedule calendar. The Robot Schedule calendar used is the one currently assigned to the message set. For more information about Robot Schedule calendars, see the *Robot Schedule User Guide*.

There are several Robot Schedule calendar variables that you can use:

- CALENDAR
- WORKDAY
- DAY
- LASTDAY
- WEEKNO
- DAYMTH

These variables are described in the following sections.

CALENDAR—Robot Schedule Calendar Name

The **CALENDAR** variable contains the name of the Robot Schedule calendar that's in effect. If a message set doesn't have a specific Robot Schedule calendar assigned to it, the STANDARD calendar is used. If the OPAL code is used by more than one message set and the message sets use different calendars, use this OPAL variable to check which calendar is in effect.

Value: Robot Schedule calendar name (up to 10 characters). To pick from a list of Robot Schedule calendars, position the cursor in the Operation Value field and press function key 4.

Example: The following condition is true if the Robot Schedule STANDARD calendar is in effect.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | CALENDAR | EQ | STANDARD |

WORKDAY—Is it a Working Day?

The **WORKDAY** variable contains a true value (YES) if today is a working day as defined by the Robot Schedule calendar currently in effect. Otherwise, it contains a NO value.

Value: YES or NO (or Y or N).

Example: The following condition is true if today is not a working day.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | WORKDAY | EQ | NO |
| THEN | | PAGE2WAY | OPERATOR |
| END | | | |

DAY—Day of the Week

The **DAY** variable gets the number (1-7) of the current day of the week (Monday, Tuesday, and so on). The current day of the week is determined by the Robot Schedule calendar currently in effect.

Value: A number where Monday is 1, Tuesday is 2, and so forth up to 7 for Sunday.

Example: The following condition is true if today is a Friday.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | DAY | EQ | 5 |

LASTDAY—Last Date in the Month

The **LASTDAY** variable contains the date of the last day of the current month as defined by the Robot Schedule calendar currently in effect.

Value: A six-digit date.

Note: If you compare the LASTDAY variable with a date in a Robot reserved command variable, make sure that the date in the Robot variable is in yymmdd (year, month, day) format.

Example: The following condition is true if today is the last day of the month.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | LASTDAY | EQ | SYSDATE |

WEEKNO—Week Number

The **WEEKNO** variable contains the week number of today within the current month. The first seven days of the month are week 1, the next seven days are week 2, and so forth. The

previous month end is defined by the Robot Schedule calendar. For example, if a fiscal month ends August 28, the week number for August 29 is 1.

Value: A one-digit number, where 1 is the first week in the month, 2 is the second, and so forth.

Example: The following condition is true if today is after the first week of the month.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | WEEKNO | GT | 1 |

DAYMTH—Day Number in the Month

The **DAYMTH** variable contains the day number of today within the current month as defined by the Robot Schedule calendar currently in effect.

Value: A one- or two-digit day number counting from the beginning of the month. The end of the month is defined in the Robot Schedule calendar used. For example, if a fiscal month ends August 28, the day number for August 29 is 1.

Example: The following condition is true if today is after the tenth day of the month.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | DAYMTH | GT | 10 |

Use Information About the Message

The OPAL variables described in the following sections contain information about the message being processed. The variables contain the date and time when the message was sent; the message group the message is assigned to; the message type, severity, and status codes; the job and program that sent the message; the message center it went to; and the message identifier and message text.

There are a number of message information variables that you can use:

- MSGDATE
- MSGGRP
- MSGTIME
- MSGTYPE
- MSGSEV

- MSGSTS
- JOB
- JOBNUMBER
- USER
- FROMPGM
- CENTER
- MSGID
- TEXT
- TXT2NDLVL
- USRTEXT
- USRTEXT2

These variables are described in the following sections.

MSGDATE—Message Date

The **MSGDATE** variable contains the date on which the message was sent.

Value: A six-digit number representing the date in the system format. For example, if the system format is year, month, day (yymmdd), 150106 is January 6, 2015.

Note: If you compare the MSGDATE variable with a date in a Robot reserved command variable, make sure that the date in the Robot variable is in yymmdd (year, month, day) format.

Example: The following condition is true if the message was sent before today.

| Logic Operand | Variable | Operation | Operation Values |
|------------------|----------|-----------|---------------------|
| IF | MSGDATE | LT | SYSDATE |

MSGGRP—Message Group

The **MSGGRP** variable contains the name of the message group that the message is assigned to.

Value: The message group name (up to 10 characters).

Example: The following condition is true if the message is in the message group Console.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | MSGGRP | EQ | Console |

MSGTIME—Message Time

The **MSGTIME** variable contains the time at which the message was sent.

Value: A six-digit number representing the time as hours, minutes, and seconds on a 24-hour clock. For example, 003000 is 30 minutes after midnight.

Example: The following conditions are true if the message was sent after 8:00 a.m. but before 5:00 p.m.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | MSGTIME | GT | 080000 |
| AND | MSGTIME | LT | 170000 |

MSGTYPE—Message Type

The **MSGTYPE** variable contains the type of the message, as follows:

- **01** Completion
- **02** Diagnostic
- **04** Information
- **05** Inquiry
- **08** Request
- **10** Request with prompting
- **14** Notify
- **15** Escape
- **21** Reply, not checked for validity
- **22** Reply, already checked for validity
- **23** Reply, message default used
- **24** Reply, system default used
- **25** Reply, from System Reply List
- **LI** Logged informational messages
- **LQ** Logged inquiry messages

Value: Any of the two-character type codes listed above.

Example: The following condition is true if the message type is Information.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | MSGTYPE | EQ | 04 |

MSGSEV—Message Severity

The **MSGSEV** variable contains the standard message severity code for the message. The severity codes include:

- **00** Information
- **10** Warning
- **20** Error
- **30** Severe Error
- **40** Abnormal End of Program or Function
- **50** Abnormal End of Job
- **60** System Status
- **70** Device Integrity
- **80** System Alert
- **90** System Integrity
- **99** Action

Value: Severity code 00-99.

Example: This condition is true if the message severity is less than Error.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | MSGSEV | LT | 20 |

MSGSTS—Message Status

The **MSGSTS** variable contains a Robot Console message status code:

- **WT** Message is waiting for a reply.
- **RS** A response required message is waiting for a reply.
- **MC** Message received a reply from a message center.

- **MD** Message received a reply from the message default.
- **MS** Message received a reply from a message set.
- **RL** Message received a reply from the system reply list.
- **IN** Information message.
- **OC** Message received a reply from outside of Robot Console.
- **RA** Reply received from Robot Alert.

Value: Any of the two-character status codes listed above.

Example: The following condition is true if the message is a response required message waiting for a reply.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | MSGSTS | EQ | RS |

JOB—Job Name that Sent Message

The **JOB** variable contains the name of the job that sent the message.

Value: The job name (up to 10 characters).

Example: The following condition is true if the name of the job that sent the message is BACK123.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | JOB | EQ | BACK123 |

JOBNUMBER—Job Number that Sent Message

The **JOBNUMBER** variable contains the number of the job that sent the message.

Value: The job number (six numeric characters).

Example: The following condition is true if the number of the job that sent the message is 472802.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|-----------|-----------|------------------|
| IF | JOBNUMBER | EQ | 472802 |

USER—User Whose Job Sent Message

The **USER** variable contains the name of the user profile that ran the job that sent the message.

Value: The user profile name (up to 10 characters).

Example: The following condition is true if the user profile whose job sent the message is BILL.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | USER | EQ | BILL |

FROMPGM—Program that Sent Message

The **FROMPGM** variable contains the name of the program that sent the message.

Value: The program name (up to 10 characters).

Example: The following condition is true if the program that sent the message is SALUPD.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | FROMPGM | EQ | SALUPD |

CENTER—Message Center

The **CENTER** variable contains the name of the message center where the message was originally sent. A message is sent to a message center if its message queue is assigned to the message center. If a message group has been assigned to the message, the message center is the name of the center specified by the message group. However, if *MSGQ is specified on the message group, the message center is the one the message queue is assigned to.

Value: The name of a message center defined in Robot Console (up to 10 characters). To pick from a list of message centers, position the cursor in the Operation Value field and press function key 4.

Example: The following condition is true if the message was sent to message center BETSY.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | CENTER | EQ | BETSY |

MSGID—Message Identifier

The **MSGID** variable contains the message identifier (ID) of the message being processed. If the message set processes a range of messages, you can determine which message is being processed by checking the message identifier.

Value: The seven-character message identifier. The first three characters are a code beginning with a letter.

Example: The following condition is true if the message ID is CPF3701.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | MSGID | EQ | CPF3701 |

TEXT—First-Level Message Text

The **TEXT** variable contains the complete first-level message text, including the data substituted for each variable in the message text. If the message set processes a range of messages, you can determine which message is being processed by checking the message text.

Value: The characters to be found in the text.

Example: The following condition is true if the first-level message text contains the characters 'General Ledger'.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | TEXT | CT | General Ledger |

TXT2NDLVL—Second-Level Message Text

The **TXT2NDLVL** variable is similar to the TEXT variable (above), but it contains the first 1000 characters of the second-level message text, including the data substituted for each variable in the message text.

Value: The characters to be found in the text.

Example: The following condition is true if the second-level message text contains the characters 'Accounts Payable'.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|-----------|-----------|------------------|
| IF | TXT2NDLVL | CT | Accounts Payable |

USRTEXT—Customized Message Text

The **USRTEXT** variable allows you to customize a message by concatenating several values. For example, you can concatenate your own text, message text, and other OPAL variable values and send them as a single message using the EXECUTE operation. You can concatenate the values using either the BCAT or CAT operations.

Value: Any combination of values up to 512 characters. User text must be enclosed in single quotes.

Example: The following statement combines the text “This is the message.” with the message ID, the actual text of the IBM i message, the job name, and the job number, and sends them as a message to a specified user.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|--------------------------|
| | USRTEXT | CHGTO | 'This is the message.' |
| | USRTEXT | BCAT | MSGID |
| | USRTEXT | CAT | ':' |
| | USRTEXT | BCAT | TEXT |
| | USRTEXT | BCAT | 'From job/job#:' |
| | USRTEXT | BCAT | JOB |
| | USRTEXT | CAT | '/' |
| | USRTEXT | CAT | JOBNUMBER |
| | | EXECUTE | SENDMSG MSG (USRTEXT)... |

USRTEXT2—Customized Message Text

The **USRTEXT2** variable is similar to the USRTEXT variable (above), except that it provides a longer text field that can hold up to 1000 characters.

Value: Any combination of values up to 1000 characters. User text must be enclosed in single quotes.

For an example, see USRTEXT (above).

Use the Message History

You can use OPAL variables to reference information about earlier messages sent by this job. To provide you with information about earlier messages, Robot Console builds a table containing information about the five previous messages sent by the job. The table might look like the following:

| Order (<i>n</i>) | Message ID (PRIORMSG <i>n</i>) | Reply (PRIORRPY <i>n</i>) | Message Group and Set (PRIORMRS <i>n</i>) |
|-----------------------|------------------------------------|-------------------------------|--|
| 1 | CPF1032 | G | Console, SET01032 |
| 2 | CPF0343 | | *NONE, SET00343 |
| 3 | CPF1032 | G | Console, SET01032 |
| 4 | CPF3394 | | Console, SET01032 |
| 5 | CPF1032 | G | Console, SET01032 |

Each entry in the table can contain the message ID, the first 32 characters of the message reply, and the message group and message set names of the message set that ran for the message.

You can use the following OPAL variables to reference the information in the table:

- PRIORMSG*n*
- PRIORRPY*n*
- PRIORMRS*n*

For example, for the message two messages back, PRIORMSG2 contains the message ID, PRIORRPY2 contains the message reply (if any), and PRIORMRS2 contains the message group and message set names of the message set that ran for the message (if any).

These variables are described in the following sections.

PRIORMSG n —Earlier Messages in the Job

The **PRIORMSG n** variables contain the message identifiers of the five previous messages sent by the job. PRIORMSG1 references the immediately preceding message, PRIORMSG2 references the message that's two messages back, PRIORMSG3 the message that's three messages back, and so forth up to PRIORMSG5.

Value: The message identifier (ID). Or, specify BLANK to check if this is the first message sent by the job.

Example: The following condition is true if the message sent by the job before the message currently being processed was CPF3272. The second condition is true if the ID for the current message is the same as the ID of the immediately preceding message (the message data could differ).

| Logic Operand | Variable | Operation | Operation Values |
|---------------|-----------|-----------|------------------|
| IF | PRIORMSG1 | EQ | CPF3272 |
| OR | PRIORMSG1 | EQ | MSGID |

PRIORRPY n —Earlier Message Replies in the Job

The **PRIORRPY n** variables contain the first 32 characters of the replies to the five previous messages sent by the job. PRIORRPY1 contains the reply to the last message, PRIORRPY2 contains the reply that's two messages back, PRIORRPY3 contains the reply that's three messages back, and so forth, up to PRIORRPY5.

Note: The PRIORRPY n variable is blank if the message did not require a reply. For example, in the [message history table](#) shown in the Message History section, variable PRIORRPY2 is blank because message number 2 did not require a reply.

Value: The characters to be found in the first 32 characters of the reply.

Example: The following condition is true if the reply to the previous message sent by the job was G.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|-----------|-----------|------------------|
| IF | PRIORRPY1 | EQ | G |

PRIORMRS n —Earlier Message Sets that Ran in the Job

The **PRIORMRS_n** variables contain the message group and message set names of the message sets that ran for the five previous messages sent by the job. PRIORMRS1 contains the message group and message set names of the message set that ran for the last message, PRIORMRS2 contains the message group and message set names of the message set that ran for the message two messages back, and so forth, up to PRIORMRS5.

Note: The PRIORMRS_n variable is blank if a message set did not run for the message. For example, in the [message history table](#) shown in the Use the Message History section, variable PRIORMRS4 is blank because no message set ran for message number 4.

Value: The name of the message group and the message set (up to 21 characters). You can enter up to 10 characters for the message group name, up to 10 characters for the message set name, and a comma to separate the names. If the message group is *NONE (the message is not assigned to a message group), enter *NONE plus the message set name.

Example: The following condition is true if message set SET05817 in message group Console ran for the last message sent by the job.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|-----------|-----------|------------------|
| IF | PRIORMRS1 | EQ | Console,SET05817 |

Use Message Repeats

The OPAL variables described in the following sections contain information about the number of times a message was sent.

There are four message repeats variables that you can use:

- REPEAT
- REPEATJOB
- REPEATMSG
- REPEATSET

These variables are described in the following sections.

REPEAT – Message Repeat Count

Must match Job Sending Message, Message Queue and Library, Message ID, Message File and Library, and Message Data.

The **REPEAT** variable contains the number of times this message has repeated in the time limit. The time limit is set by the message set or by the system default value. To be

considered a repeat, the messages must match exactly, that is, both the message ID and the message data must match an earlier message.

Use the REPEAT variable to limit the number of times the same message is processed. For example, suppose a line failure message is received and a message set varies the line off and then on. If the line fails to become active, the same line failure message is received again. The message set again varies the line off and on. This loop could continue indefinitely. To prevent that, you should limit the number of times the same message is processed.

The time limit for counting repetitions should be one that is appropriate for the message being processed. Device messages always contain the same job name and number until the next IPL. Thus, the time limit helps determine if processing is actually in a loop.

Value: The number of times the same message has been processed (0-99).

Example: The following condition is true if the same message has been processed less than seven times.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | REPEAT | LT | 7 |

REPEATJOB – Message Repeat Count

Must match Job Sending Message, Message Queue and Library, Message ID, and Message File and Library.

The **REPEATJOB** variable contains the number of times the same message has repeated in the time limit. The time limit is set by the message set or by the system default value. The REPEATJOB variable differs from the REPEAT variable in that it doesn't look at the message data in each message.

Use the REPEATJOB variable when the messages being processed contain IBM error log identifiers, which are used by IBM to work with problem log messages. The error log identifier in the message is incremented each time a message is sent, causing the message data to differ.

Value: The number of times the same message has been processed (0-999).

Example: The following condition is true if the same message ID has occurred less than eight times.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|-----------|-----------|------------------|
| IF | REPEATJOB | LT | 8 |

REPEATMSG – Message Repeat Count

Must match Message ID and Message File and Library.

The **REPEATMSG** variable contains the number of times the same message has repeated in the time limit. The time limit is set by the message set or by the system default value. The REPEATMSG variable differs from the REPEAT and REPEATJOB variables; it doesn't look at the job sending the message, the message queue and library, or the message data in each message.

Value: The number of times the same message has been processed (0-999).

Example: The following condition is true if the same message has been processed less than five times.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|-----------|-----------|------------------|
| IF | REPEATMSG | LT | 5 |

REPEATSET – Message Repeat Count

Must match Message Set Group and Message Set Name.

The **REPEATSET** variable contains the number of times the same message has repeated in the time limit. The time limit is set by the message set or by the system default value. The REPEATSET variable counts the number of times the current message set has been used in the specified time limit.

Value: The number of times the same message has been processed (0-999).

Example: The following condition is true if the same message has been processed less than ten times.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|-----------|-----------|------------------|
| IF | REPEATSET | LT | 10 |

Note: The REPEATSET variable has been changed to provide new functionality. Therefore, if you have existing OPAL code that uses REPEATSET, it won't function the same as it did previously. OPAL message sets should be evaluated, and you should use the OPAL variable that best meets your needs.

Process the Reply

When a reply is received to an inquiry message, the Reply OPAL code, if any, is executed to process the reply. Use the OPAL variables in the Reply OPAL code for information about the reply.

There are several reply processing variables:

- RPYTEXT
- RPYUSER
- RPYJOB
- RPYJOBNBR
- RPYCENTER

These variables are described in the following sections.

RPYTEXT—Reply Text

The **RPYTEXT** variable contains the complete text (up to 132 characters) of the message reply.

Value: The characters to be found in the reply.

Example: The following condition is true if the reply contains the character G.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | RPYTEXT | CT | G |

RPYUSER—User Who Replied

The **RPYUSER** variable contains the name of the user profile that replied to the message.

Value: The user profile name (up to ten characters).

Example: The following condition is true if the user who replied was MARKJ.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | RPYUSER | EQ | MARKJ |

RPYJOB—Job that Replied

The **RPYJOB** variable contains the name of the job that replied to the message.

Value: The job name (up to ten characters).

Example: The following condition is true if the name of the job that replied to the message was BACKUP1.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | RPYJOB | EQ | BACKUP1 |

RPYJOBNBR—Job Number that Replied

The **RPYJOBNBR** variable contains the number of the job that replied to the message.

Value: The six-digit job number.

Example: The following condition is true if the job number that replied to this message in the job was 321789.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|-----------|-----------|------------------|
| IF | RPYJOBNBR | EQ | 321789 |

RPYCENTER—Message Center that Replied

The **RPYCENTER** variable contains the name of the message center that replied to this message.

Value: The message center name (up to ten characters). To pick from a list of message centers, position the cursor in the Operation Values field and press function key 4.

Example: The following condition is true if the message center that replied to the message was WAREHOUSE.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|-----------|-----------|------------------|
| IF | RPYCENTER | EQ | WAREHOUSE |

Assign Values to User Variables

OPAL provides fifteen user variables that don't contain a value until you assign one. You can assign a value to a user variable using the CHGTO operation and later reference the variable in a condition or operation.

You can use these variables:

- USRFLD n (1-5)
- USRFLG n (1-5)
- USRNBR n (1-5)
- NO
- YES

These variables are described in the following sections.

USRFLD n —Character Variables

The **USRFLD n** variables are five user variables that can contain up to ten characters each. The first variable is referenced as USRFLD1, the second as USRFLD2, and so forth, up to USRFLD5.

Value: Up to ten characters.

Example: The following condition is true if user field 1 contains the characters 'LINE'. The characters must be enclosed in single quotation marks because LINE is the name of an OPAL variable.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | USRFLD1 | EQ | 'LINE' |

USRFLG n —Flag Variables

The **USRFLG n** variables are five user variables that can contain one character each. The first field is referenced as USRFLG1, the second as USRFLG2, and so forth, up to USRFLG5.

Value: One character.

Example: The following condition is true if user flag 1 contains the character Y.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | USRFLG1 | EQ | Y |

USRNBR n —Numeric Variables

The **USRNBR n** variables are five user variables that can contain a number in format (5,0). The first field is referenced as USRNBR1, the second as USRNBR2, and so forth, up to USRNBR5.

Value: Number in format (5,0).

Example: The following condition is true if USRNBR1 is less than 7.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | USRNBR1 | LT | 7 |

NO—False Logical Comparison for USRFLG n Variables

The **NO** variable contains a false logical comparison for the USRFLG n variables. Use this variable when you are specifying a false relationship between the NO variable and the value in the Operation Values field.

Value: One character

Example: The following statement is true if NO is equal to the value in the variable USRFLG1.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | NO | EQ | USRFLG1 |

YES—False Logical Comparison for USRFLG n Variables

The **YES** variable contains a true logical comparison for the USRFLG n variables. Use this variable when you are specifying a true relationship between the YES variable and the value in the Operation Values field.

Value: One character

Example: The following statement is true if YES is equal to the value in the variable USRFLG2.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | YES | EQ | USRFLG2 |

Using OPAL Message Data Variables

Up to 600 characters of data can be passed with a message. Using the first 30 message variables, the message description can break that data into fields and define variables in the message text (&1, &2, and so forth), which are substituted by the data fields.

There are several categories of variables for message data:

- Message data
- Devices
- Resources
- SNADS Distribution

The following sections cover the variables that are in each of these categories.

Use the Message Data

OPAL lets you use the message data. You can reference the entire data buffer or an individual variable. You can reference the first 30 message variables by number or, in some cases, by name.

The message data variables:

- DATA
- VARn
- FILE
- LIBRARY
- MEMBER
- NUMBER
- SUBSYSTEM
- SYSTEM

- PROGRAM
- JOURNAL

These variables are described in the following sections.

DATA—Message Data

The **DATA** variable contains the data that came with the message. Up to 256 characters of data can be passed with a message. The data replaces the variables in the message text.

Value: Characters to be found in the data.

Example: The following condition is true if the message data contains the characters PAYMAST.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | DATA | CT | PAYMAST |

VARn—Message Variables by Number

The **VARn** variables contain the data used to replace variables in the message text. VAR1 contains the value for message data field 1 as defined in the message description, VAR2 contains the value for message data field 2, and so forth up to VAR30.

Value: Up to 20 characters to be found in the data or matched to the data.

Example: The following condition is true if the value for variable 1 in the message is SALUPD.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | VAR1 | EQ | SALUPD |

FILE—File Name in Message

If the message text contains the word File followed by a variable, the OPAL variable **FILE** contains the file name from the message.

Value: The file name (up to 10 characters).

Example: The following condition is true if the file name is RBT323.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | FILE | EQ | RBT323 |

LIBRARY—Library Name in Message

If the message text contains the word Library followed by a variable, the OPAL variable **LIBRARY** contains the library name from the message.

Value: The library name (up to 10 characters).

Example: The following condition is true if the library name is RBTLIB.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | LIBRARY | EQ | RBTLIB |

MEMBER—Member Name in Message

If the message text contains the word Member followed by a variable, the OPAL variable **MEMBER** contains the member name from the message.

Value: The member name (up to 10 characters).

Example: The following condition is true if the member name is RBTHelp.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | MEMBER | EQ | RBTHelp |

NUMBER—Network File Number in Message

If the message text contains the word Number followed by a variable, the OPAL variable **NUMBER** contains the network file number from the message.

Value: The network file number (six numeric characters).

Example: The following condition is true if the number is 264776.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | NUMBER | EQ | 264776 |

SUBSYSTEM—Subsystem Name in Message

If the message text contains the word Subsystem followed by a variable, the OPAL variable **SUBSYSTEM** contains the subsystem name from the message.

Value: The subsystem name (up to 10 characters).

Example: The following condition is true if the subsystem name is RBTSLEEPER.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|-----------|-----------|------------------|
| IF | SUBSYSTEM | EQ | RBTSLEEPER |

SYSTEM—System Name in Message

If the message text contains the word System followed by a variable, the OPAL variable **SYSTEM** contains the system name from the message.

Value: The system name (up to 10 characters).

Example: The following condition is true if the system name is D10.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | SYSTEM | EQ | D10 |

PROGRAM—Program Name in Message

If the message text contains the word Program followed by a variable, the OPAL variable **PROGRAM** contains the program name from the message.

Value: The program name (up to 10 characters).

Example: The following condition is true if the program name is RBTSEC.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | PROGRAM | EQ | RBTSEC |

JOURNAL—Journal Name in Message

If the message text contains the word **Journal** followed by a variable, the OPAL variable **JOURNAL** contains the journal name from the message.

Value: The journal name (up to 10 characters).

Example: The following condition is true if the journal name is JOURNAL1.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | JOURNAL | EQ | JOURNAL1 |

Use Devices from the Message Data

OPAL lets you use device variables from the message data. You can reference an individual device variable by name.

The device variables:

- LINE
- CONTROLLER
- DEVICE
- WRITER
- FORMTYPE

These variables are described in the following sections.

LINE—Line Name in Message

If the message text contains the word **Line** followed by a variable, the OPAL variable **LINE** contains the line name from the message.

Value: The line name (up to 10 characters).

Example: The following condition is true if the line name is DULUTH.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | LINE | EQ | DULUTH |

CONTROLLER—Controller Name in Message

If the message text contains the word Controller followed by a variable, the OPAL variable **CONTROLLER** contains the controller name from the message.

Value: The controller name (up to 10 characters).

Example: The following condition is true if the controller name is TAPECTRL.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|------------|-----------|------------------|
| IF | CONTROLLER | EQ | TAPECTRL |

DEVICE—Device Name in Message

If the message text contains the word Device followed by a variable, the OPAL variable **DEVICE** contains the device name from the message.

Value: The device name (up to 10 characters).

Example: The following condition is true if the device name is DISK01.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | DEVICE | EQ | DISK01 |

WRITER—Writer Name in Message

If the message text contains the word Writer followed by a variable, the OPAL variable **WRITER** contains the writer name from the message.

Value: The writer name (up to 10 characters).

Example: The following condition is true if the writer name is WRITER01.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | WRITER | EQ | WRITER01 |

FORMTYPE—Form Type Name in Message

If the message text contains the word Formtype followed by a variable, the OPAL variable **FORMTYPE** contains the form type name from the message.

Value: the form type name (up to 10 characters).

Example: The following condition is true if the form type name is INVOICE01.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | FORMTYPE | EQ | INVOICE01 |

Use SNADS Distribution from the Message Data

OPAL lets you use SNADS (Systems Network Architecture Distribution Services) distribution variables from the message data. You can reference an individual SNADS variable by name.

The SNADS variables:

- DISTRIBQ
- SENDER
- SENDERNBR
- SENDERUSR
- RECEIVER
- RCVNBR
- RCVUSR
- ROUTER
- ROUTERNBR
- ROUTERUSR

These variables are described in the following sections.

DISTRIBQ—Distribution Queue Name in Message

If the message text contains the word Distribq followed by a variable, the OPAL variable **DISTRIBQ** contains the distribution queue name from the message.

Value: The distribution queue name (up to 10 characters).

Example: The following condition is true if the distribution queue name is B10.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | DISTRIBQ | EQ | B10 |

SENDER—Sender Name in Message

If the message text contains the word **Sender** followed by a variable, the OPAL variable **SENDER** contains the sender job name from the message.

Value: The sender job name (up to 10 characters).

Example: The following condition is true if the sender job is SENDER01.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | SENDER | EQ | SENDER01 |

SENDERNBR—Sender Number in Message

If the message text contains the word **Sendernbr** followed by a variable, the OPAL variable **SENDERNBR** contains the sender job number from the message.

Value: The sender job number (six numeric characters).

Example: The following condition is true if the sender job number is 231552.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|-----------|-----------|------------------|
| IF | SENDERNBR | EQ | 231552 |

SENDERUSR—Sender User in Message

If the message text contains the word **Senderusr** followed by a variable, the OPAL variable **SENDERUSR** contains the user profile name for the sender job from the message.

Value: The user profile name (up to 10 characters).

Example: The following condition is true if the user profile for the sender job is KAREN.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|-----------|-----------|------------------|
| IF | SENDERUSR | EQ | KAREN |

RECEIVER—Receiver Name in Message

If the message text contains the word **Receiver** followed by a variable, the OPAL variable **RECEIVER** contains the receiver name from the message.

Value: The receiver job name (up to 10 characters).

Example: The following condition is true if the receiver job is RECVR01.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | RECEIVER | EQ | RECVR01 |

RCVNBR—Receiver Number in Message

If the message text contains the word Rcvnbr followed by a variable, the OPAL variable **RCVNBR** contains the receiver job number from the message.

Value: The receiver job number (six numeric characters).

Example: The following condition is true if the receiver job number is 231552.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | RCVNBR | EQ | 231552 |

RCVUSR—Receiver User in Message

If the message text contains the word Rcvusr followed by a variable, the OPAL variable **RCVUSR** contains the user profile name for the receiver job from the message.

Value: The user profile name (up to 10 characters).

Example: The following condition is true if the user profile for the receiver job is KAREN.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | RCVUSR | EQ | KAREN |

ROUTER—Router Name in Message

If the message text contains the word Router followed by a variable, the OPAL variable **ROUTER** contains the router job name from the message.

Value: The router job name (up to 10 characters).

Example: The following condition is true if the router job is ROUTER01.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | ROUTER | EQ | ROUTER01 |

ROUTERNBR—Router Number in Message

If the message text contains the word Routernbr followed by a variable, the OPAL variable **ROUTERNBR** contains the router job number from the message.

Value: The router job number (six numeric characters).

Example: The following condition is true if the router job number is 231552.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|-----------|-----------|------------------|
| IF | ROUTERNBR | EQ | 231552 |

ROUTERUSR—Router User in Message

If the message text contains the word Routerusr followed by a variable, the OPAL variable **ROUTERUSR** contains the user profile name for the router job from the message.

Value: The user profile name (up to 10 characters).

Example: The following condition is true if the user profile for the router job is KAREN.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|-----------|-----------|------------------|
| IF | ROUTERUSR | EQ | KAREN |

Use Resources from the Message Data

OPAL lets you use resource variables from the message data. You can reference an individual resource variable by name.

The resource variables:

- RESOURCE
- RSCID
- STATUS

- OBJTYP
- TYPE

These variables are described in the following sections.

RESOURCE—Resource Name in Message

If the message text contains the word Resource followed by a variable, the OPAL variable **RESOURCE** contains the resource name from the message.

Value: The resource name (up to 10 characters).

Example: The following condition is true if the resource name is NTSERVER.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | RESOURCE | EQ | NTSERVER |

RCSID—Resource ID in Message

If the message text contains the word RCSID followed by a variable, the OPAL variable **RSCID** contains the resource ID from the message.

Value: The resource ID (nine numeric characters). You must enter nine characters; enter leading zeroes as needed.

Example: The following condition is true if the resource ID is 000000158.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | RSCID | EQ | 000000158 |

STATUS—Resource Status in Message

If the message text contains the word Status followed by a variable, the OPAL variable **STATUS** contains the resource status value from the message.

Value: The status of the resource (up to 10 characters).

Example: The following condition is true if the resource status is RELEASED.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | STATUS | EQ | RELEASED |

OBJTYP—Extended Resource Object Type in Message

If the message text contains the word Objtyp followed by a variable, the OPAL variable **OBJTYP** contains the IBM object type from the message. The OBJTYP variable extends the standard resource definition for the object.

Value: The object type (up to 10 characters).

Example: The following condition is true if the object type is *DTAARA.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | OBJTYP | EQ | *DTAARA |

TYPE—Resource Type in Message

If the message text contains the word Type followed by a variable, the OPAL variable **TYPE** contains the resource type from the message.

Value: The resource type (up to 10 characters).

Example: The following condition is true if the resource type is SERVER.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | TYPE | EQ | SERVER |

Working with Application Log Monitor Variables

OPAL provides variables for use with application log monitors. This includes eighteen that are 128 characters long and that don't contain a value until you assign one. You can assign a value to a user variable using the CHGTO operation and later reference the variable in a condition or operation.

You can use these variables:

- FILTERNAME
- LOGDATA
- STRINGnn

These variables are described in the following sections.

FILTERNAME—Name of the Filter Triggering the Event

The **FILTERNAME** variable is the name of the filter that triggered the event that the application log monitor was checking for.

Value: The filter name (up to 20 characters).

Example: The following condition is true if the filter name is 'Invalid License'.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|------------|-----------|------------------|
| IF | FILTERNAME | EQ | Invalid License |

LOGDATA—Log Data Being Monitored

The **LOGDATA** variable contains up to 256 characters of data from the log file being monitored.

Value: The characters to be found in the text.

Example: The following condition is true if the monitored log contains the characters Firewall.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | LOGDATA | CT | Firewall |

STRINGnn—Strings from the Monitored Log

The eighteen **STRINGnn** variables can each contain a string of up to 128 characters of data from the log file that's being monitored. The first string is referenced as STRING01, the second as STRING02, and so forth, up to STRING18. **Note:** This variable can be used outside of application log monitoring. It can be used wherever you need more than the 20 characters that the [VARn variable](#) allows.

Value: Up to 128 characters to be found in the text.

Example: The following condition is true if STRING01 contains the characters 'General Ledger'.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | STRING01 | CT | General Ledger |

Comparisons

When you specify a condition, you must specify a comparison in the Operation field. The comparison specifies a relationship between the value of the OPAL variable in the Variable field and the value in the Operation Value field. If the relationship exists, the condition is true.

For example, the following condition specifies the greater than comparison (GT) in the Operation field. For the condition to be true, the system time (specified by the OPAL variable SYSTIME) must be greater than the operation value (190000).

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | SYSTIME | GT | 190000 |

The following sections describe the comparisons that are available in OPAL.

Equality Comparisons

For equality, the comparisons you can specify in the Operation field are as follows:

| | |
|-----------|--|
| EQ | Equals; the values match exactly. |
| NE | Not equals; the values do not match exactly. |

Order Comparisons

The following comparisons are for comparing the order of the values. For numeric values, a numeric comparison is done. For alphanumeric values, an alphanumeric comparison is done, character by character, from left to right. For example, the numeric value 12 is greater than both 1 and 2. But, the alphanumeric value 12 is greater than 1, but less than 2.

| | |
|-----------|--------------|
| GT | Greater than |
|-----------|--------------|

| | |
|-----------|--------------------------|
| GE | Greater than or equal to |
| LT | Less than |
| LE | Less than or equal to |

Containing Comparisons

The following two comparisons compare the characters to see if they contain or don't contain the values you're looking for. These comparisons are case-sensitive. When it tests the condition, OPAL looks for the same sequence of characters from the operation value in the OPAL variable value. For example, the character sequence SALE is contained in SALES, NEWSALE, and OLDSALES, but it's not contained in SALUPD.

| | |
|-----------|---|
| CT | Contains; the character sequence from the operation value is also in the OPAL variable value. |
| DC | Doesn't contain; the character sequence from the operation value is not in the OPAL variable value. |

OPAL Table Comparisons

These two comparisons are used with OPAL tables. The data in the variable field is compared to all elements in the specified OPAL table. The match must be exact.

| | |
|-------------------|---|
| INTABLE | The variable data exists in the OPAL table. |
| NOTINTABLE | The variable data does not exist in the OPAL table. |

Message Table Comparisons

These two comparisons are used with message tables. The data in the variable field is compared to all message IDs in the specified message table. The match must be exact.

| | |
|-------------------|--|
| INMSGTABLE | The variable data exists in the message table. |
|-------------------|--|

| | |
|----------------------|--|
| NOTINMSGTABLE | The variable data does not exist in the message table. |
|----------------------|--|

Operations

The purpose of the OPAL code in a message set is to tell Robot Console what to do to process a message . The OPAL statements that tell Robot Console what to do are called operation statements.

OPAL code must include at least one operation statement. Any other statements in the OPAL code just determine which operation statements are performed.

The general syntax of an operation statement is as follows:

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| optional | optional | required | optional |

When you code an operation statement, you always fill in the Operation field; you provide an Operation Value if one is needed by the operation. The CHGTO, ADD, and SUB operations require an OPAL variable name in the Variable field. Usually, the Logic Operand field is left blank, unless the operation is part of a THEN statement.

For example, the following statement uses only the Operation field.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| | | DEFAULT | |

Most operation statements use the Operation and Operation Value fields, as follows:

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| | | ENTER | G |

The following CHGTO operation also uses the Variable field:

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| | USRNBR1 | CHGTO | 7 |

The pages that follow describe what you can enter in the Operation field.

Reply to the Message

You can specify an unconditional reply to the message as part of the Unconditional Instructions for the message set. Or, you can specify a conditional reply using OPAL code. The OPAL code can specify conditions that must be met before the reply statement is processed. After a reply is sent, processing continues with the After-Reply OPAL code, if any.

The available operations for replying to a message:

- DEFAULT
- ENTER

These operations are described in the following sections.

DEFAULT—Default Reply

The **DEFAULT** operation enters the default reply to the message.

To see what the default reply is for a message, use the Display Message File option and display the message attributes for the message. The default reply is in the list of attributes.

Operation value: None.

Example: The following operation sends the default message reply.

| Logic Operand | Variable | Operation | Operation Values |
|------------------|----------|-----------|---------------------|
| | | DEFAULT | |

ENTER—Enter a Reply

The ENTER operation responds to the message with the reply entered in the Operation Value field.

Operation value: Any valid reply to the message. To pick from a list of valid replies (as defined in the message description), position the cursor in the Operation Value field and press function key 4.

Example: The following operation answers the message with the reply G.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| | | ENTER | G |

Wait Awhile

Use these operations to tell Robot Console to wait awhile before processing the next OPAL statement.

The available operations for waiting:

- RPYWITHIN
- DELAY

These operations are described in the following sections.

RPYWITHIN—Time Limit for Reply

The **RPYWITHIN** operation has Robot Console wait the specified number of seconds for a message reply. If a reply is received in the time limit, processing continues with the After-Reply OPAL code (if any). If no reply is received within the time limit, processing continues with the next OPAL statement.

Operation value: The number of seconds Robot Console should wait for a reply.

Example: The following statement waits 300 seconds for a reply.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| | | RPYWITHIN | 300 |

DELAY—Delay Processing

The **DELAY** operation has Robot Console wait the specified number of seconds before it processes the next statement. For example, a delay might be needed for an earlier command to finish processing.

Do not use DELAY if you're waiting for a reply; the message set cannot receive a reply while it is in the delay state. If you're waiting for a reply, use RPYWITHIN instead of DELAY; RPYWITHIN can receive a reply while it's waiting.

Operation value: The number of seconds Robot Console should wait.

Example: The following statement delays processing for 60 seconds.

| Logic Operand | Variable | Operation | Operation Values |
|------------------|----------|-----------|---------------------|
| | | DELAY | 60 |

Change the Message

Use these operations to suppress the message or to change the message so it requires a response, becomes a system alert message, or notifies a user; or to remove a message.

The available operations for changing messages:

- SUPPRESS
- RESPOND
- SNDAlert
- NOTIFY
- NOTIFYL
- REMOVE

These operations are described in the following sections.

SUPPRESS—Do Not Display the Message

The **SUPPRESS** operation prevents display of the message on the original message center. If the message has been copied to other message centers, the copies are not suppressed. Once a message has been suppressed, no other escalation occurs.

You can suppress a message unconditionally using an option on the Unconditional Instructions panel. Use a SUPPRESS operation in the OPAL code to suppress the message only when conditions are met. When the message is suppressed in the OPAL code, a history record still exists for the message; when the message is unconditionally suppressed, no history record is written.

You cannot suppress a message that the user must respond to. You cannot suppress a message range as an operation value.

Operation value: None.

Example: The following statement prevents display of the message being processed.

| Logic Operand | Variable | Operation | Operation Values |
|------------------|----------|-----------|---------------------|
| SUPPRESS | | | |

RESPOND—Require a Response to the Message

The **RESPOND** operation changes an informational message so that a response is required. Any response is valid.

Change important informational messages to require a response. This brings the powerful notification procedures of Robot Console into play so that important events (such as disk failure) will be brought to the attention of the IT staff. You can change the message unconditionally with an option on the Unconditional Instructions panel or change it conditionally in the OPAL code.

The message history records who responds to the message, and the date and time of the response.

Operation value: None.

Example: The following statement changes the message so it requires a response.

| Logic Operand | Variable | Operation | Operation Values |
|------------------|----------|-----------|---------------------|
| RESPOND | | | |

SNDAAlert—Change the Message to a System Alert

The **SNDAAlert** operation converts the message being processed into a system alert.

Alerts appear on the IBM Work with Alerts display (WRKALR). If the IBM i is part of a network of non-IBM i systems, the message may appear on the Alerts display on other systems. An alert is intended to notify the network operator of an actual or impending loss of a resource.

To see the message text on the IBM i, enter **option 8** for the alert on the Work with Alerts panel, and then display the second page of the Display Alert Detail panel. **Note:** Alerts will not appear on the Work with Alerts panel if the Alert Status network attribute is set to *OFF. Enter the DSPNETA command to check the value of the network attribute.

Operation value: Up to eight characters to appear as the Resource Name on the Work with Alerts display.

Example: The following statement changes the message into a system alert. PAYROLL appears as the resource name for the alert.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| | | SNDA!ert | PAYROLL |

NOTIFY—Notify a User

The **NOTIFY** operation displays a pop-up window to notify a user that a message has arrived on a message center. The window displays the message text and gives the user the option of answering the message without having to go to the message center. Informational messages can be deleted directly from the pop-up window. **Note:** To notify a list instead of a single user, see the next section.

Operation value: The user profile of the person who should be notified of a message (up to 10 characters).

Example: The following statement notifies user DAVEJ that a message has arrived on a message center.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| | | NOTIFY | DAVEJ |

NOTIFYL—Notify a List

The **NOTIFYL** operation allows you to specify a notification list to be notified that a message has arrived on a message center. **Note:** To notify a single user instead of a list, see the previous section.

Operation value: The name of the notification list that contains the users who should be notified of a message.

Example: The following statement notifies the users in the notification list OPSLIST that a message has arrived on a message center.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| | | NOTIFYL | OPSLIST |

REMOVE—Remove a Message

The **REMOVE** operation removes a message from a message center, but retains message history.

Operation value: None.

Example: The following statement removes the message being processed from the message center.

| Logic Operand | Variable | Operation | Operation Values |
|------------------|----------|-----------|---------------------|
| | | REMOVE | |

Send the Message Elsewhere

Use these operations to send a message to another message center or to another message queue.

Note: If your system is a node in an IBM i network controlled by Robot Network, you can redirect or copy the message to a message center on another system in the network. To do so, just specify the message center and the system name, separated by a comma.

The available operations for sending messages elsewhere:

- REDIRECT
- COPY
- SENDQ

These operations are described in the following sections.

REDIRECT—Send Message to Another Message Center

The **REDIRECT** operation removes the message from the current message center and sends it to another message center. Use this operation to pass the message on to someone else to answer. The RPYWITHIN operation can set a time limit for a reply before you redirect the message.

Operation value: The message center name. To redirect to another system in your network controlled by Robot Network, specify the message center and the system name separated by a comma.

Examples: The following statement sends the message to message center WAREHOUSE.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| | | REDIRECT | WAREHOUSE |

The following statement sends the message to message center JIM on the system DALLAS.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| | | REDIRECT | JIM,DALLAS |

COPY—Copy to Another Message Center

The **COPY** operation sends a copy of the message to another message center. Use this operation if you want someone else to see the message.

Notes:

- You cannot copy inquiry or response-required messages until after they've been answered.
- Copying a message does not remove it from your message center.

Operation value: The message center name. To copy to another system in your network controlled by Robot Network, specify the message center and the system name separated by a comma.

Examples: The following statement sends a copy of the message to message center BETTY.


| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| | | COPY | BETTY |

The following statement sends a copy of the message to message center FRED on the system SEATTLE.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| | | COPY | FRED,SEATTLE |


SENDQ—Send to Another Message Queue

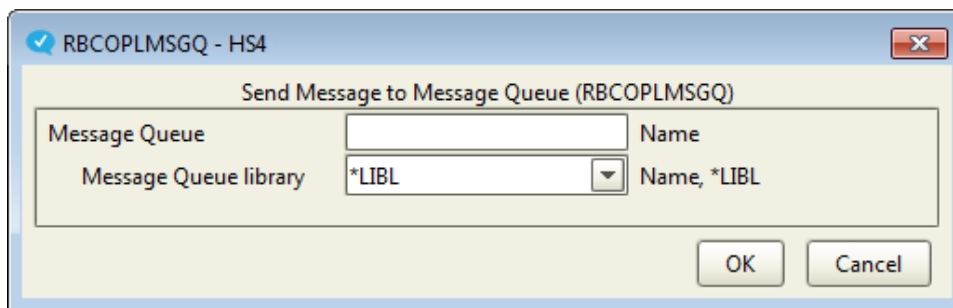
The **SENDQ** operation sends the message to a message queue. Use this operation to send the message to a personal message queue or to another queue that Robot Console doesn't monitor.

Operation value: Position the cursor in the Operation Values field and press function key 4 (or click the Finder icon  in Robot Console Explorer). A prompt screen appears as shown below. Enter the message queue name and library.

Example: The following statement sends the message to message queue QSYSOPR.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|---------------------------------|
| | | SENDQ | Press F4 to see command. |

After you press F4 (or the Finder icon ), enter the message queue and library to which the message should be sent.




Send a New Message

Use this operation to send a new message to a message center. See the following section for details.

SENDMC—Send a New Message to Another Message Center

The **SENDMC** operation sends a new message to a message center. Use this operation to inform someone of the message processing that the OPAL code has done.

Operation value: Position the cursor in the Operation Values field and press function key 4 (or click the Finder icon  in the Robot Console Explorer). A prompt screen is displayed as shown below. Enter the message center name and the message to be sent. You can also require a response to the message.

Examples: The following statement sends a message to another message center.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|--------------------------------|
| | | SENDMC | Press F4 to see command |

There are three pieces of information that you need to enter:

- Enter the **Message Center Name**.
- Enter the **Message Text** you want to send.
- Select whether or not to require a response.

Send a Message to a Device

If you have Robot Alert installed, you can use these operations to send a message to a Robot Alert device or broadcast list.


The available operations for sending messages with Robot Alert:

- PAGE
- PAGE2WAY

These operations are described in the following sections.

PAGE—Send a One-Way Message to a Robot Alert Device

If you have Robot Alert installed, the **PAGE** operation executes the Robot Alert command that sends a message to a specified device or broadcast list. For more information, see the *Robot Alert User Guide*. **Note:** This is strictly a one-way operation, even if you specify a two-way device. For two-way communication, see PAGE2WAY in the next section.

Operation value: Position the cursor in the Operation Values field and press function key 4 (or click the Finder icon  in Robot Console Explorer). If you have Robot Alert installed, the

prompt screen for the RBASNDMSG command appears. Enter the device name and the message to be sent.


Examples: The following statement sends a message to a Robot Alert device or broadcast list.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|--------------------------------|
| | | PAGE | Press F4 to see command |

Enter the message text and the name of the device or broadcast list. Then, set or change the additional parameters. On the IBM i, press **F10** to see the additional parameters.

PAGE2WAY—Send a Two-Way Message to a Device

If Robot Alert version 5 or later is installed on your system, and you have a two-way device, the **PAGE2WAY** operation sends a message to a specified device, receives a reply from the user, and updates Robot Console. The message contains the message text and the valid replies. You cannot send alternative message text when you use this operation. For more information about sending two-way messages, see the *Robot Alert User Guide*.

Operation value: Position the cursor in the Operation Values field and press function key 4 (or click the Finder icon  in Robot Console Explorer) to select from a list of two-way devices defined to Robot Alert. You also can enter the name of a device in the field; if the name you enter is not a two-way device, an error message displays.

Examples: The following statement sends a two-way message to the device ONCALL.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| | | PAGE2WAY | ONCALL |


Execute a Command

Use this operation to execute any IBM i command. See the following section for details.

EXECUTE – Execute any Command

The **EXECUTE** operation can execute any IBM i command.

Operation value: Type the command to be executed in the Operation Values field.

In the Robot Console Explorer, type the command or press F4 (or click the Finder icon ) to select a command. Then, enter the parameters in the window that opens. Click **OK** to enter the command under Operation Values.


On the IBM i, if more space is needed, press F4 to display the Extended Command Entry panel. To display the command prompt screen, type the command name and then press F4. Enter the command parameter values. Press Enter to enter the command on the Extended Command Entry panel.

Example: The following statement executes a VRYCFG command.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|-------------------------|
| | | EXECUTE | VRYCFG CFGOBJ (LINE)... |

You can enter OPAL variables, such as LINE, as parameter values.

Robot Reserved Command Variables

If you have Robot Schedule installed, you can press F7 (or click the Reserved Command Variable Finder icon  in Robot Console Explorer) to see a list of reserved command variables. You can use any of these variables as a parameter value in the command. The current value of the variable is substituted when the command is executed.

Reserved command variables can contain global system values such as operator on duty, shift hours, and so forth. You can change the value of a reserved command variable in

reaction to a message using the command RBTCHGRSV. For more information, see the *Robot Schedule User Guide*.

Call a User Program

Use these operations to call user programs from the OPAL code. You can pass either the message data or the reply data to the program.

The available operations for calling a user program:

- CALLCPV3
- CALLCP
- CALLRPY

These operations are described in the following sections.

CALLCPV3—Pass the Message Data to a Program

The **CALLCPV3** operation calls a user program and passes it the message data in the parameter RBCMEPV3. **Note:** This operation passes the message data using the Robot Console 3 message data format. The Robot Console CALLCP operation is converted to CALLCPV3 during conversion to version 4.

Operation value: The program name or the library name/program name.

Example: The following statement passes the message data to program MYPRG in the current library list.

| Logic Operand | Variable | Operation | Operation Values |
|------------------|----------|-----------|---------------------|
| | | CALLCPV3 | MYPRG |

RBCMEPV3 Parameter

The user program must define the RBCMEPV3 parameter as an external data structure as follows:

| | | | |
|---|--------|--------|--------|
| D | RBCMEP | E DS | 1024 |
| C | | *ENTRY | PLIST |
| C | | PARM | RBCMEP |

The RBCMEPV3 format is as follows. This is the Robot Console 3 format as used in Robot Console 4. In general, the values are those described for OPAL variables.

| Field | Type | Size | Dec | In | Out | Text |
|--------|------|------|-----|-----|-----|---------------------------|
| E#MSKY | A | 4 | | 1 | 4 | Message reference key |
| E#MSGQ | A | 10 | | 5 | 14 | Message queue name |
| E#MSQL | A | 10 | | 15 | 24 | Message queue library |
| E#MSAL | A | 9 | | 25 | 33 | Message alert option |
| E#JBNM | A | 10 | | 34 | 43 | Message from job name |
| E#JBNO | A | 6 | | 44 | 49 | Message from job number |
| E#MSPG | A | 10 | | 50 | 59 | Message from program name |
| E#USER | A | 10 | | 60 | 69 | Message from user name |
| E#RPNO | P | 3 | 0 | 70 | 71 | Repeat number |
| E#MSD | P | 6 | 0 | 72 | 75 | Date of message |
| E#MSSV | A | 2 | | 76 | 77 | Message severity |
| E#MTYP | A | 2 | | 78 | 79 | Message type |
| E#TMSG | P | 6 | 0 | 80 | 83 | Time of message |
| E#WRTR | A | 10 | | 84 | 93 | Writer |
| E#SYSN | A | 10 | | 94 | 103 | System name |
| E#FORM | A | 10 | | 104 | 113 | Form type |
| E#SUBS | A | 10 | | 114 | 123 | Subsystem |
| E#DVNM | A | 10 | | 124 | 133 | Device name |
| E#LNNM | A | 10 | | 134 | 143 | Line name |
| E#CNTR | A | 10 | | 144 | 153 | Controller |

| Field | Type | Size | Dec | In | Out | Text |
|--------|------|------|-----|-----|-----|-------------------------|
| E#MSDT | A | 256 | | 154 | 409 | Message data or message |
| E#DAYN | A | 1 | | 410 | 410 | Day number of week |
| E#WKNO | A | 1 | | 411 | 411 | Week number |
| E#DAYM | A | 2 | | 412 | 413 | Day of month |
| E#WORK | A | 1 | | 414 | 414 | Work day |
| E#SNDR | A | 10 | | 415 | 424 | Sender |
| E#SUSR | A | 10 | | 425 | 434 | Sender user |
| E#SNBR | A | 6 | | 435 | 440 | Sender job number |
| E#RTR | A | 10 | | 441 | 450 | Router name |
| E#RUSR | A | 10 | | 451 | 460 | Router user |
| E#RNBR | A | 6 | | 461 | 466 | Router number |
| E#RECV | A | 10 | | 467 | 476 | Receiver |
| E#RCUR | A | 10 | | 477 | 486 | Receiver user |
| E#RCJN | A | 6 | | 487 | 492 | Receiver number |
| E#FLNM | A | 10 | | 493 | 502 | File name |
| E#MBR | A | 10 | | 503 | 512 | Member name |
| E#LIBR | A | 10 | | 513 | 522 | Library name |
| E#DBRQ | A | 10 | | 523 | 532 | Distribution queue |
| E#JRNM | A | 10 | | 533 | 542 | Journal name |
| E#VR1 | A | 10 | | 543 | 552 | Variable 1 |
| E#VR2 | A | 10 | | 553 | 562 | Variable 2 |
| E#VR3 | A | 10 | | 563 | 572 | Variable 3 |
| E#VR4 | A | 10 | | 573 | 582 | Variable 4 |
| E#VR5 | A | 10 | | 583 | 592 | Variable 5 |
| E#VR6 | A | 10 | | 593 | 602 | Variable 6 |
| E#VR7 | A | 10 | | 603 | 612 | Variable 7 |

| Field | Type | Size | Dec | In | Out | Text |
|--------|------|------|-----|-----|-----|-------------------------------------|
| E#VR8 | A | 10 | | 613 | 622 | Variable 8 |
| E#VR9 | A | 10 | | 623 | 632 | Variable 9 |
| E#VR10 | A | 10 | | 633 | 642 | Variable 10 |
| E#VR11 | A | 10 | | 643 | 652 | Variable 11 |
| E#PBRP | A | 1 | | 653 | 653 | Create program problem report |
| E#MFLB | A | 10 | | 654 | 663 | Message file library |
| E#MSFL | A | 10 | | 664 | 673 | Message file |
| E#MSID | A | 7 | | 674 | 680 | Message identification |
| E#SDAT | P | 6 | 0 | 681 | 684 | Reserved |
| E#STIM | P | 6 | 0 | 685 | 688 | Reserved |
| E#MSLN | P | 5 | 0 | 689 | 691 | Message received length |
| E#MSPS | A | 4 | | 692 | 695 | Message from pgm stmt num |
| E#MSTS | A | 2 | | 696 | 697 | Message status |
| E#RESP | A | 1 | | 698 | 698 | Acknowledge required? |
| E#SYSC | A | 10 | | 699 | 708 | Created by system |
| E#MCNM | A | 10 | | 709 | 718 | Message center name |
| E#RBCL | A | 10 | | 719 | 728 | Calendar name |
| E#LSDT | P | 6 | 0 | 729 | 732 | Last day of month |
| E#FNBR | P | 6 | | 733 | 738 | Network file number |
| E#MDTS | P | 6 | 0 | 739 | 742 | Message date system format |

CALLCP—Pass the Message Data to a Program

The **CALLCP** operation calls a user program and passes it the message data in the parameter RBCMEP. **Note:** This operation passes the message data using the Robot Console 4 message data format.

Operation value: The program name or the library name/program name.

Example: The following statement passes the message data to program MYPRG in the current library list.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| | | CALLCP | MYPRG |

RBCMEP Parameter

The user program must define the RBCMEP parameter as an external data structure as follows:

| | | | |
|---|--------|--------|--------|
| D | RBCMEP | E DS | 4119 |
| C | | *ENTRY | PLIST |
| C | | PARM | RBCMEP |

The RBCMEP format is as follows. In general, the values are those described for OPAL variables.

| Field | Type | Size | Dec | In | Out | Text |
|--------|------|------|-----|----|-----|-----------------------|
| E#MSKY | A | 4 | | 1 | 4 | Message reference key |
| E#MSGQ | A | 10 | | 5 | 14 | Message queue name |
| E#MSQL | A | 10 | | 15 | 24 | Message queue library |
| E#MSAL | A | 9 | | 25 | 33 | Message alert option |
| E#JBNM | A | 10 | | 34 | 43 | Message from job name |

| Field | Type | Size | Dec | In | Out | Text |
|--------|------|------|-----|-----|-----|---------------------------|
| E#JBNO | A | 6 | | 44 | 49 | Message from job number |
| E#MSPG | A | 10 | | 50 | 59 | Message from program name |
| E#USER | A | 10 | | 60 | 69 | Message from user name |
| E#MSD | P | 6 | 0 | 70 | 73 | Date of message |
| E#MSSV | A | 2 | | 74 | 75 | Message severity |
| E#MTYP | A | 2 | | 76 | 77 | Message type |
| E#TMSG | P | 6 | 0 | 78 | 81 | Time of message |
| E#MMSG | P | 6 | 0 | 82 | 85 | Message milliseconds |
| E#WRTR | A | 10 | | 86 | 95 | Writer |
| E#SYSN | A | 10 | | 96 | 105 | System name |
| E#FORM | A | 10 | | 106 | 115 | Form type |
| E#SUBS | A | 10 | | 116 | 125 | Subsystem |
| E#DVNM | A | 10 | | 126 | 135 | Device name |
| E#LNNM | A | 10 | | 136 | 145 | Line name |
| E#CNTR | A | 10 | | 146 | 155 | Controller |
| E#DAYN | A | 1 | | 156 | 156 | Day number of week |
| E#WKNO | A | 1 | | 157 | 157 | Week number |
| E#DAYM | A | 2 | | 158 | 159 | Day of month |
| E#WORK | A | 1 | | 160 | 160 | Work day |
| E#SNDR | A | 10 | | 161 | 170 | Sender |
| E#SUSR | A | 10 | | 171 | 180 | Sender user |
| E#SNBR | A | 6 | | 181 | 186 | Sender job number |
| E#RTR | A | 10 | | 187 | 196 | Router name |
| E#RUSR | A | 10 | | 197 | 206 | Router user |

| Field | Type | Size | Dec | In | Out | Text |
|--------|------|------|-----|-----|-----|--------------------|
| E#RNBR | A | 6 | | 207 | 212 | Router number |
| E#RECV | A | 10 | | 213 | 222 | Receiver |
| E#RCUR | A | 10 | | 223 | 232 | Receiver user |
| E#RCJN | A | 6 | | 233 | 238 | Receiver number |
| E#FLNM | A | 10 | | 239 | 248 | File name |
| E#MBR | A | 10 | | 249 | 258 | Member name |
| E#LIBR | A | 10 | | 259 | 268 | Library name |
| E#DBRQ | A | 10 | | 269 | 278 | Distribution queue |
| E#JRNM | A | 10 | | 279 | 288 | Journal name |
| E#VR1 | A | 20 | | 289 | 308 | Variable 1 |
| E#VR2 | A | 20 | | 309 | 328 | Variable 2 |
| E#VR3 | A | 20 | | 329 | 348 | Variable 3 |
| E#VR4 | A | 20 | | 349 | 368 | Variable 4 |
| E#VR5 | A | 20 | | 369 | 388 | Variable 5 |
| E#VR6 | A | 20 | | 389 | 408 | Variable 6 |
| E#VR7 | A | 20 | | 409 | 428 | Variable 7 |
| E#VR8 | A | 20 | | 429 | 448 | Variable 8 |
| E#VR9 | A | 20 | | 449 | 468 | Variable 9 |
| E#VR10 | A | 20 | | 469 | 488 | Variable 10 |
| E#VR11 | A | 20 | | 489 | 508 | Variable 11 |
| E#VR12 | A | 20 | | 509 | 528 | Variable 12 |
| E#VR13 | A | 20 | | 529 | 548 | Variable 13 |
| E#VR14 | A | 20 | | 549 | 568 | Variable 14 |
| E#VR15 | A | 20 | | 569 | 588 | Variable 15 |
| E#VR16 | A | 20 | | 589 | 608 | Variable 16 |
| E#VR17 | A | 20 | | 609 | 628 | Variable 17 |
| E#VR18 | A | 20 | | 629 | 648 | Variable 18 |
| E#VR19 | A | 20 | | 649 | 668 | Variable 19 |

| Field | Type | Size | Dec | In | Out | Text |
|--------|------|------|-----|-----|-----|-------------------------------|
| E#VR20 | A | 20 | | 669 | 688 | Variable 20 |
| E#VR21 | A | 20 | | 689 | 708 | Variable 21 |
| E#VR22 | A | 20 | | 709 | 728 | Variable 22 |
| E#VR23 | A | 20 | | 729 | 748 | Variable 23 |
| E#VR24 | A | 20 | | 749 | 768 | Variable 24 |
| E#VR25 | A | 20 | | 769 | 788 | Variable 25 |
| E#VR26 | A | 20 | | 789 | 808 | Variable 26 |
| E#VR27 | A | 20 | | 809 | 828 | Variable 27 |
| E#VR28 | A | 20 | | 829 | 848 | Variable 28 |
| E#VR29 | A | 20 | | 849 | 868 | Variable 29 |
| E#VR30 | A | 20 | | 869 | 888 | Variable 30 |
| E#PBRP | A | 1 | | 889 | 889 | Create program problem report |
| E#MFLB | A | 10 | | 890 | 899 | Message file library |
| E#MSFL | A | 10 | | 900 | 909 | Message file |
| E#MSID | A | 7 | | 910 | 916 | Message identification |
| E#SDAT | P | 6 | 0 | 917 | 920 | Reserved |
| E#STIM | P | 6 | 0 | 921 | 924 | Reserved |
| E#MSLN | P | 5 | 0 | 925 | 927 | Message received length |
| E#MSPS | A | 4 | | 928 | 931 | Message from pgm stmt num |
| E#MSTS | A | 2 | | 932 | 933 | Message status |
| E#RESP | A | 1 | | 934 | 934 | Acknowledge required? |
| E#SYSC | A | 10 | | 935 | 944 | Created by system |
| E#MCNM | A | 10 | | 945 | 954 | Message center name |
| E#RBCL | A | 10 | | 955 | 964 | ROBOT calendar name |

| Field | Type | Size | Dec | In | Out | Text |
|----------------|------|------|-----|------|------|----------------------------|
| E#LSDT | P | 6 | 0 | 965 | 968 | ROBOT last day of month |
| E#FNBR | P | 6 | | 969 | 974 | Network file number |
| E#MDTS | P | 6 | 0 | 975 | 978 | Message date system format |
| E#MPGM | A | 10 | | 979 | 988 | Program |
| E#AMGRP | A | 10 | | 989 | 998 | Assigned message group |
| E#EMDTA | A | 3072 | | 999 | 4070 | Extended message data |
| E#RSCID | A | 9 | | 4071 | 4079 | Resource ID |
| E#STATUS | A | 10 | | 4080 | 4089 | Expected/actual status |
| E#RESOURC E | A | 10 | | 4090 | 4099 | Resource name |
| E#OBJTYP | A | 10 | | 4100 | 4109 | Extended type |
| E#TYPE | A | 10 | | 4110 | 4119 | Resource Type |

CALLRPY—Pass the Reply Data to a Program

The **CALLRPY** operation calls a user program and passes it the reply data in the parameter RBCQRM. Because a reply must be available to pass to the program, only the OPAL code that processes the message reply (the After-Reply OPAL code) can use the CALLRPY operation.

Operation value: The program name or the library name/program name.

Example: The following statement passes the message reply to program RPYPRG in the current library list.

| Logic Operand | Variable | Operation | Operation Values |
|------------------|----------|-----------|---------------------|
| | | CALLRPY | MYPRG |

RBCQRM Parameter

The user program must define the RBCQRM parameter as follows:

| | | | |
|---|--------|--------|--------|
| D | RBCQRM | E DS | 512 |
| C | | *ENTRY | PLIST |
| C | | PARM | RBCQRM |


The RBCQRM format is as follows.

| Field | Type | Size | Dec | In | Out | Text |
|--------|------|------|-----|-----|-----|-----------------------------|
| R#MSKY | A | 4 | | 1 | 4 | Message reference key |
| R#MSID | A | 7 | | 5 | 11 | Message identification |
| R#MSRP | A | 132 | | 12 | 143 | Message reply |
| R#MSGQ | A | 10 | | 144 | 153 | Message queue name |
| R#MSQL | A | 10 | | 154 | 163 | Message queue library |
| R#MSTS | A | 2 | | 164 | 165 | Message status |
| R#ANMC | A | 10 | | 166 | 175 | Answered by message center |
| R#ANMS | P | 5 | 0 | 176 | 178 | Answered by message set |
| R#MCNM | A | 10 | | 179 | 188 | Message center name |
| R#MAJB | A | 10 | | 189 | 198 | Message answered by job |
| R#MAUS | A | 10 | | 199 | 208 | Message answered by user |
| R#MAJN | A | 6 | | 209 | 214 | Message answered by job no. |

| Field | Type | Size | Dec | In | Out | Text |
|--------|------|------|-----|-----|-----|----------------------------|
| R#MSNO | P | 5 | 0 | 215 | 217 | Message set number |
| R#RPDT | P | 6 | 0 | 218 | 221 | Reply date |
| R#RPTM | P | 6 | 0 | 222 | 225 | Reply time |
| R#RPOS | A | 10 | | 226 | 235 | Reserved |
| R#RPUS | A | 10 | | 236 | 245 | Reserved |
| R#RPSY | A | 10 | | 246 | 255 | Reserved |
| R#MDTS | P | 6 | 0 | 256 | 259 | Message date system format |
| R#TMSG | P | 6 | 0 | 260 | 263 | Time of message |
| R#WAIT | P | 5 | 0 | 264 | 266 | Suspend wait time |
| R#RMTR | A | 1 | | 267 | 267 | Remote reply? |
| R#MMSG | P | 6 | 0 | 268 | 271 | Message milliseconds |
| R#MGRP | A | 10 | | 272 | 281 | Message group name |
| R#MSNM | A | 10 | | 282 | 291 | Message set name |

Run a Robot Schedule Job

If you have Robot Schedule installed, you can use the **RBTBCHUPD** command to change a Robot Schedule job record. Use the EXECUTE operation to execute the RBTBCHUPD command. You can change the job schedule and job setup options and pass command variable values and the current LDA as needed. For more information, see the *Robot Schedule User Guide*.

Operation value: Type the following command in the Operation Values field: ROBOTLIB/RBTBCHUPD and **F4** (or click the Finder icon  in the Robot Console Explorer). The prompt screen for the RBTBCHUPD command is displayed. Enter the command parameter values(see the image below from the Explorer).

Example: The following operation executes the Robot Schedule RBTBCHUPD command.


| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|--------------------------------|
| | | EXECUTE | ROBOTLIB/RBTBCHUPD... |
| | | | Press F4 to see command |

The ROBOT Job Number identifies the Robot Schedule job record. All the other parameters change options in the Robot Schedule job record. For more information about each option, see the *Robot Schedule User Guide*.

Run a Robot Schedule Reactive Job

If you have Robot Schedule installed, the **SNDRBTDTA** operation can send notification of the message or of another event to Robot Schedule. The notification satisfies a prerequisite for a reactive job. For more information on reactive jobs, see the *Robot Schedule User Guide*.

The SNDRBTDTA operation sends a completion status to Robot Schedule. Robot Schedule records that status in the job prerequisite list. If all prerequisites for the job have been met, the reactive job can run.

Operation value: Position the cursor in the Operation Value field and press **F4** (or click the Finder icon  in the Robot Console Explorer). The prompt screen for the SNDRBTDTA command is displayed (see the image below from the Explorer). Enter the job name and completion status for a user job that has already been entered in the prerequisite list of a Robot Schedule reactive job.

Example: The following statement sends notification of the message to Robot Schedule.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|--------------------------------|
| | | SNDRBTDTA | Press F4 to see command |

Enter the name of a user job that's already on a Robot Schedule prerequisite list. Also, enter the completion status that was entered for the user job in the prerequisite list.

Send an SNMP Trap

Robot Console supports SNMP (Simple Network Management Protocol) traps. A “trap” is a software packet containing data (such as system messages) that's stored in a predefined data structure for interpretation and use by enterprise monitor software. Robot Console can create an SNMP trap when it receives a message on the IBM i, or when a message has received a reply. The trap is then passed to the enterprise monitor, which collects it, interprets its contents, and displays it in a format determined by the monitoring site. The information allows the enterprise monitor administrator to take care of the message or pass the information to another source. The communication is one-way—events are sent to the enterprise monitor, but replies cannot be returned to Robot Console.

The available operations for SNMP traps:

- SNDSNMPPMSG
- SNDSNMPPRY

These operations are described in the following sections.

SNDSNMPPMSG—Send Notification of a Message

The **SNDSNMPPMSG** operation sends notification of a message to an SNMP trap manager or enterprise monitor. Use this operation when Robot Console auto-creates a message set.

Operation Value: None

Example: The following statement sends notification of a message to the trap manager.

| Logic Operand | Variable | Operation | Operation Values |
|------------------|----------|-------------|---------------------|
| | | SNDSNMPPMSG | |

SNDSNMPPRY—Send Message Reply Information

The **SNDSNMPPRY** operation sends message reply information to the SNMP trap manager when you create a message set to forward the reply. The SNDSNMPPRY operation is used in the Reply OPAL.

Operation Value: None

Example: The following statement sends message reply information to the trap manager.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|------------|------------------|
| | | SNDSNMPPRY | |

Stop Processing

Use these operations to stop processing the message. You can choose to have the operator answer the message, to end all processing of the message, or to terminate the job that sent the message.

The available operations to stop the processing of a message:

- OPERATOR
- QUIT
- TERMINATE

These operations are described in the following sections.

OPERATOR—Let the Operator Answer

The **OPERATOR** operation lets whoever has the message on their message center answer the message. OPAL processing stops and the procedures for waiting for a reply (auto-paging and/or auto-redirect) are performed.

Operation value: None.

Example: The following statement stops message processing and waits for the operator to reply to the message.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| | | OPERATOR | |

QUIT—Stop All Processing

The **QUIT** operation stops all processing of the message including redirection and paging procedures.

Operation value: None.

Example: The following statement stops all processing of the message.

| Logic Operand | Variable | Operation | Operation Values |
|------------------|----------|-----------|---------------------|
| | | QUIT | |

TERMINATE—End the Job

The **TERMINATE** operation terminates the job that sent the message. It also stops all processing of the message.

Operation value: None.

Example: The following statement ends the job that sent the message.

| Logic Operand | Variable | Operation | Operation Values |
|------------------|----------|-----------|---------------------|
| | | TERMINATE | |

Change OPAL Variable Values

Use these operations to change the values of OPAL variables, especially the fifteen user variables. The user variables can be used to control program logic, such as a counter for a WHILE loop, or to hold operation values to be used later.

These operations are described in the following sections.

CHGTO—Change the Value of a User Variable

Use the **CHGTO** operation to assign a value to a user variable. Enter the name of the user variable (USRFLD n , USRFLG n , or USRNBR n , where n is 1 to 5) in the Variable field.

Operation value: The value to be assigned to the user variable. For USRFLD n , specify up to ten characters; for USRFLG n , specify one character; for USRNBR n , specify a numeric value (for example: 5,0).

Example: The following statement changes the USRFLD1 value to DAVEJ.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| | USRFLD1 | CHGTO | DAVEJ |

ADD—Add to a Numeric User Variable

Use the **ADD** operation to add to the numeric user variable specified in the Variable field (USRNBR n , where n is 1 to 5).

Operation value: The number to be added to the numeric value already in the USRNBR n variable. For USRNBR n , specify a numeric value in the format (15, 5); for USRCNT n specify a numeric value in the format (5, 0). The number to be added can also be specified by a USRNBR n variable.

Example: The following statement adds 1 to the value in the USRNBR1 field.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| | USRNBR1 | ADD | 1 |

Example: The following statement adds the value in the USRNBR1 variable to USRNBR2.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| | USRNBR2 | ADD | USRNBR1 |

SUB—Subtract from a Numeric User Variable

Use the **SUB** operation to subtract from the numeric user variable specified in the Variable field (USRNBR n , where n is 1 to 5).

Operation value: The number to be subtracted from the numeric value already in the USRNBR n variable. The number to be subtracted can also be specified by a USRNBR n variable.

Example: The following statement subtracts 1 from the value in the USRNBR1 variable.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| | USRNBR1 | SUB | 1 |

Example: The following statement subtracts the value in the USRNBR1 variable from USRNBR2.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| | USRNBR2 | SUB | USRNBR1 |

CAT—Concatenate Two Values

Use the **CAT** operation to concatenate two values. The CAT operation removes any trailing blanks from the first string and concatenates the second string, with no added blanks. This is similar to the CL *TCAT command.

Operation value: The value to be concatenated to the user variable. The value specified also can be a variable.

Example: The following statement concatenates XYZ to the value in USRFLD1.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| | USRFLD1 | CAT | XYZ |

BCAT—Add a Blank Before Concatenation

The **BCAT** operation works just like CAT, except it adds a space before concatenating the values.

Operation value: The value to be concatenated to the user variable. The value specified also can be a variable.

Example: The following statement adds a blank and XYZ to the end of the value in USRFLD1.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| | USRFLD1 | BCAT | XYZ |

Go to Tagged Location

Use the TAG operation to assign a name to a location in the program, and then use a GOTO operation to continue processing at that location. The * operation lets you add comments to the program.

The available operations for tagging a location:

- TAG
- GOTO
- *

These operations are described in the following sections.

TAG—Tag a Program Location

The **TAG** operation assigns a name to a location in the program. A GOTO operation can then specify the tag name to go to the tagged location.

Operation value: The name for the location.

Example: The following statement assigns the name BEGIN to this location in the OPAL code.

| Logic Operand | Variable | Operation | Operation Values |
|------------------|----------|-----------|---------------------|
| | | TAG | BEGIN |

GOTO—Go to Tagged Location

The **GOTO** operation transfers processing to the location specified by the tag name. When a GOTO is performed, processing immediately “jumps” to the tagged location. Thus, the next statement processed is the statement that follows the TAG operation.

Operation value: The name specified on a TAG operation in the code.

Example: The following statement continues processing at the TAG statement that contains the name BEGIN.

| Logic Operand | Variable | Operation | Operation Values |
|------------------|----------|-----------|---------------------|
| | | GOTO | BEGIN |

*—Add a Comment

The * (asterisk) operation indicates that the statement is a comment used to document the processing performed by the program. You can enter text in the Operation Value field to describe what the code does, or you can leave the field blank to improve the readability of the code.

Operation value: Any characters.

Note: A comment cannot appear between an IF and an END statement.

Example: The following statement inserts a blank line in the code for readability.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| | | * | |

GOTO Example

The following is an example of OPAL code that uses the TAG, GOTO, and * operations. The code repeats until either Bill or Rob replies to the message.

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|--------------------------------|
| | | TAG | BEGIN |
| | | * | |
| | | * | Bill gets 5 minutes to answer. |
| | | REDIRECT | BILL |
| | | RPYWITHIN | 300 |
| | | * | |
| | | * | Rob gets 5 minutes to answer. |
| | | REDIRECT | ROB |
| | | RPYWITHIN | 300 |
| | | * | |
| | | * | Go back to Bill. |
| | | GOTO | BEGIN |

Operation Values

Most OPAL conditions and operations require a value in the Operation Values field. The value can be represented by a constant or by an OPAL variable.

For example, if the system date is 150102, the following statements are the same except that the first uses a constant and the second uses an OPAL variable:

| Logic Operand | Variable | Operation | Operation Values |
|---------------|----------|-----------|------------------|
| IF | MSGDATE | GE | 150102 |
| IF | MSGDATE | GE | SYSDATE |

OPAL Constants

You can use a word as a constant even though it's also the name of an OPAL variable. To do so, enclose the word in single quotation marks (''). For example, LINE is an OPAL variable so, to use LINE as a constant value, you must enter 'LINE'.

The following are special OPAL constants:

| | |
|--------------|-------------------------------------|
| BLANK | Blank characters to fill the field. |
| YES | True logical value. |
| NO | False logical value. |

OPAL Variable Values

You can use any of the OPAL variables to represent a value except ACTUSR, ACTDEV, ACTCTL, or ACTLIN. For example, to represent the current time in the Operation Value field, use SYSTIME.

The following OPAL variables are available for all messages if the first- or second-level message text contains the variable name or the name has been assigned to a message variable:

| | | | | |
|-----------------|------------------|------------------|------------------|------------------|
| <u>ACTCTL</u> | <u>JOB</u> | <u>OBJTYP</u> | <u>RPYCENTER</u> | <u>TXT2NDLVL</u> |
| <u>ACTDEV</u> | <u>JOBNUMBER</u> | <u>PRIORMSGn</u> | <u>RPYJOB</u> | <u>TYPE</u> |
| <u>ACTLIN</u> | <u>JOURNAL</u> | <u>PRIORMRSn</u> | <u>RPYJOBNBR</u> | <u>USER</u> |
| <u>ACTUSR</u> | <u>LASTDAY</u> | <u>PRIORRPYn</u> | <u>RPYTEXT</u> | <u>USRFLDn</u> |
| <u>CALENDAR</u> | <u>LIBRARY</u> | <u>PROGRAM</u> | <u>RPYUSER</u> | <u>USRFLGn</u> |
| <u>CENTER</u> | <u>LINE</u> | <u>RCVNBR</u> | <u>RSCID</u> | <u>USRNBRn</u> |

| | | | | |
|-------------------|----------------|------------------|----------------------------|------------------------|
| <u>CONTROLLER</u> | <u>LOGDATA</u> | <u>RCVUSR</u> | <u>SENDER</u> | <u>USRTEXT</u> |
| <u>DATA</u> | <u>MEMBER</u> | <u>RECEIVER</u> | <u>SENDERNBR</u> | <u>USRTEXT2</u> |
| <u>DAY</u> | <u>MSGDATE</u> | <u>REPEAT</u> | <u>SENDERUSR</u> | <u>VAR_n</u> |
| <u>DAYMTH</u> | <u>MSGGRP</u> | <u>REPEATJOB</u> | <u>STATUS</u> | <u>WEEKNO</u> |
| <u>DEVICE</u> | <u>MSGID</u> | <u>REPEATSET</u> | <u>STRING_{nn}</u> | <u>WORKDAY</u> |
| <u>DISTRIBQ</u> | <u>MSGSEV</u> | <u>REPEATMSG</u> | <u>SUBSYSTEM</u> | <u>WRITER</u> |
| <u>FILE</u> | <u>MSGSTS</u> | <u>RESOURCE</u> | <u>SYSDATE</u> | |
| <u>FILTERNAME</u> | <u>MSGTIME</u> | <u>ROUTER</u> | <u>SYSTEM</u> | |
| <u>FORMTYPE</u> | <u>MSGTYPE</u> | <u>ROUTERNBR</u> | <u>SYSTIME</u> | |
| <u>FROMPGM</u> | <u>NUMBER</u> | <u>ROUTERUSR</u> | <u>TEXT</u> | |

User Variables

The OPAL variables , USRFLD_n, USRFLG_n, , and USRNBR_n are user variables. A user variable does not contain a value until the OPAL code assigns a value to it. If you specify a user variable as the operation value, the value stored in the user variable is used by the condition or operation. Do this if the operation value should vary depending on the OPAL statements executed.

For example, the following OPAL code stores in USRFLD1 the name of the person to whom the message is directed so that the Reply OPAL code can send a message to the person who replied to the message.

| Logic Operand | Variable | Operation | Operation Values |
|------------------|----------|-----------|---------------------|
| | | TAG | BEGIN |
| | USRFLD1 | CHGTO | DAVECTR |
| | | REDIRECT | USRFLD1 |
| | | RPYWITHIN | 300 |
| | USRFLD1 | CHGTO | ROHITCTR |
| | | REDIRECT | USRFLD1 |
| | | RPYWITHIN | 300 |
| | | GOTO | BEGIN |

When the reply is received, the name of the message center that replied to the message is in USRFLD1. The Reply OPAL code could then perform a SENDMC operation that sends a message to the message center in USRFLD1.

OPAL Source Code

Robot Console automatically checks the logic and syntax of your OPAL code when you leave the OPAL entry screen.

Checking the OPAL Code

You can go further in checking your OPAL code than what Robot Console does automatically. This can help you diagnose problems with your OPAL code. The method you use and the output you get is different depending on whether you're working in Robot Console Explorer (run an OPAL trace) or on the IBM i (use the RBCOPALSRC command). Both methods are described in the following sections.

OPAL Trace

To help you verify that your OPAL code is working perfectly, you can have Robot Console record an OPAL trace for a specific message set and then display the results. This shows you which OPAL operations were performed the last time the message set was used. You can even send a test message so that you don't have to wait for a real message to arrive before you can see the trace results. For details on running an OPAL trace, see the *Robot Console User Guide*.

An OPAL trace can only be performed in the Robot Console Explorer.

The RBCOPALSRC Command

Running the RBCOPALSRC command on the IBM i displays the source code generated by your OPAL code. The source code can help you troubleshoot problems. **Note:** You can only run this command on the IBM i.

Follow these steps:

1. Go to an IBM i command line and enter RBCOPALSRC.
2. Press function key 4 to see the prompt panel.

Display OPAL source code (RBCOPALSRC)

Type choices, press Enter.

| | | |
|---------------------------------|---------|------------------------|
| OPAL Message Group | *NONE | Character value, *NONE |
| OPAL Message Set Name | CJN1218 | Character value |

Bottom

F3=Exit F4=Prompt F5=Refresh F12=Cancel F13=How to use this display
F24=More keys

3. Enter the message group or message set name for which you want to display the OPAL source code. If the message is not assigned to a message group, enter *NONE.
4. The OPAL source code displays on the standard IBM Source Entry Utility (SEU) panel. You can browse through the code or print it out. If you're having trouble with your OPAL code, the Robot Technical Support consultants may ask you to print out your source code and send it to us for analysis.

```


Columns . . . :   1  71           Edit           QTEMP/OPAL00245
SEU==>                                     OPAL00245
FMT **   ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
          ***** Beginning of data *****
0001.00 /* CODE IN THE MESSAGE SET PROGRAM THAT DOES NOT HAVE A CHANGE */
0001.01 /* DATE IS DYNAMICALLY GENERATED.  OTHER CODE IS COPIED IN */
0001.02 /* FROM THE VARIOUS MODULES...(OPALDCL, OPALINIT, OPALEXIT) */
0001.03 /* */
0001.04 /* DYNAMICALLY GENERATED CODE IS NORMALLY IN LOWER CASE. */
0001.05
0001.06 /*****
0001.07 /* DECLARATION PART OF THE OPAL CODE */
0001.08 /*****
0001.09
0001.10          PGM          PARM(&CENTER &P1MSGQ &P1MSKY &P1MSNO &P1MGRP +
0001.11                      &P1MSNM &P1MSQL &P1PANM &P1PAWT &P1DIRECT +
0001.12                      &P1DQ &P1MSCN &P1MSD &P1TMSG &P1MMSG +
0001.13                      &P1DRWT &RTNVAR)
0001.14
0001.15 DCL &CENTER  *CHAR  10          /* MESSAGE CENTER */

F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve   F10=Cursor   F11=Toggle
F16=Repeat find   F17=Repeat change           F24=More keys
(C) COPYRIGHT IBM CORP. 1981, 2007.

```

Examples

OPAL Code Examples

This section contains examples of OPAL code that could be included in message sets. These are examples only. Review the code carefully; you may need to modify it for use on your system. **Note:** The images in the examples were taken on the IBM i. When entering a long command on either the IBM i or in the Robot Console Explorer, type the basic command (such as VRYCFG) and press function key 4 (or click the Finder icon  in the Explorer) to open a screen where you can enter the parameters.

Robot Console is shipped with a number of message sets already created. After you install Robot Console, these message sets can be seen in the Message Sets List view (in Robot Console Explorer) or in the list on the Maintain Message Sets panel (on the IBM i). Examine those message sets for more ideas on processing messages.

[*OPAL Example 1: Tape Device Error Can Be Recovered on page 97*](#)

[*OPAL Example 2: Message Queue Deleted or Damaged on page 98*](#)

[*OPAL Example 3: Workstation Varied Off Due to Wrong Password Entry on page 98*](#)

[*OPAL Example 4: Device Problem Messages Redirected to QSECOFR on page 99*](#)

[*OPAL Example 5: Receive Save Files and Restore Objects on page 99*](#)

[*OPAL Example 6: TCP Job Ended on page 100*](#)

[*OPAL Example 7: User Not Authorized to Change System Value on page 101*](#)

[*OPAL Example 8: Cannot Allocate Library on page 101*](#)

[*OPAL Example 9: Line Failed—Probable Local Hardware Problem on page 102*](#)

[*OPAL Example 10: Produce a Dump on page 103*](#)

[*OPAL Example 11: Answer Message and Remove from Message Center on page 103*](#)

[*OPAL Example 12: Change Informational Message to Response Required on page 104*](#)

[*OPAL Example 13: Use a Notification List on page 104*](#)

[OPAL Example 14: Use RBTBCHUPD to Run a Robot Schedule Job on page 104](#)

[OPAL Example 15: Check OPAL Table for Variable Value on page 105](#)

[OPAL Example 16: Send a System Alert and Break Message on page 105](#)

[OPAL Example 17: Change Message to Response Required on page 106](#)

[OPAL Example 18: Answer Message Depending on Device Name on page 106](#)

[OPAL Example 19: Send Message or Reset Device on page 107](#)

[OPAL Example 20: Suppress a Message or Send a Message on page 107](#)

[OPAL Example 21: End a Job on page 107](#)

[OPAL Example 22: Answer a Repeated Message on page 108](#)

[OPAL Example 23: Redirect Messages to an Active User on page 108](#)

[OPAL Example 24: Redirect Messages to HOSTCENTER on page 108](#)

OPAL Example 1: Tape Device Error Can Be Recovered

This example shows how Robot Console can handle a tape device error. When the message is received, Robot Console takes the tape device name from the message and stores it as the OPAL keyword DEVICE. OPAL code then varies off the device, waits 30 seconds, and then varies on the device.

| Logic | Operand | Variable | Operation | Operation Values | Seq |
|-------|---------|----------|-----------|-------------------------------------|-----|
| | | | * | Vary off tape device | 10 |
| | | | EXECUTE | VRYCFG CFGOBJ(DEVICE) CFGTYPE(*D... | 20 |
| | | | DELAY | 30 | 30 |
| | | | * | Vary tape device back on | 40 |
| | | | EXECUTE | VRYCFG CFGOBJ(DEVICE) CFGTYPE(*D... | 50 |
| | END | | | | 60 |

The full command for Seq 20 is as follows:

Extended Value VRYCFG CFGOBJ(DEVICE) CFGTYPE(*DEV) STATUS(*OFF)

The full command for Seq 50 is as follows:

Extended Value VRYCFG CFGOBJ(DEVICE) CFGTYPE(*DEV) STATUS(*ON)

OPAL Example 2: Message Queue Deleted or Damaged

This example shows how Robot Console handles a message about a damaged message queue. Robot Console takes the message queue and library names from the message and stores them as OPAL variables VAR1 and VAR2, respectively. OPAL code then deletes and recreates the message queue.

| Logic Operand | Variable | Operation | Operation Values | Seg |
|------------------|----------|-----------|----------------------------------|-----|
| | | * | Delete damaged message queue | 10 |
| | | EXECUTE | DLTMSGQ VAR2/VAR1 | 20 |
| | | * | Create message queue in same lib | 30 |
| | | EXECUTE | CRTMSGQ VAR2/VAR1 | 40 |
| END | | | | 50 |

OPAL Example 3: Workstation Varied Off Due to Wrong Password Entry

This example shows how Robot Console handles a message issued when the system varies off a workstation because the wrong password has been entered repeatedly. Robot Console stores the workstation name from the message (message variable 3) as OPAL variable VAR3. Then, if it's a workday, the OPAL code waits 30 minutes (if after working hours) or 10 minutes (during working hours). After the delay, it varies on the workstation. If the message is received on a non-workday, a message is sent to the specified device name and the workstation is not varied on. **Note:** The message sent by the PAGE2WAY operation includes the first-level message text. Depending on your system setup, it may also include the second-level text.

| Logic Operand | Variable | Operation | Operation Values | Seg |
|------------------|----------|-----------|-------------------------------------|-----|
| IF | WORKDAY | EQ | Y | 10 |
| IF | SYSTIME | GT | 170000 | 20 |
| OR | SYSTIME | LT | 070000 | 30 |
| | | * | If workday and after working hours | 40 |
| THEN | | DELAY | 1800 | 50 |
| | | * | If working hours, delay 10 mins | 60 |
| ELSE | | | | 70 |
| | | DELAY | 600 | 80 |
| END | | | | 90 |
| | | * | If working day, vary on workstation | 100 |
| | | EXECUTE | VRYCFG CFGOBJ(VAR3) CFGTYPE(*DEV... | 110 |
| | | * | | 120 |
| | | | More... | |
| | | * | If not workday, pager message sent | 130 |
| | | * | to warn of signon attempts | 140 |
| ELSE | | | | 150 |
| | | PAGE2WAY | DD2WAY | 160 |
| END | | | | 170 |

The full command for Seq 110 is as follows:

```
Extended Value  VRYCFG CFGOBJ(VAR3) CFGTYPE(*DEV) STATUS(*ON)
```

OPAL Example 4: Device Problem Messages Redirected to QSECOFR

This example shows how to set it up so that if user QSECOFR is signed on when a message is received, the message is redirected to message center QSECOFR.

| Logic Operand | Variable | Operation | Operation Values | Seq |
|------------------|----------|-----------|------------------------------------|-----|
| | | * | If QSECOFR is signed on | 10 |
| IF | ACTUSR | EQ | QSECOFR | 20 |
| | | * | Redirect the message to QSECOFR | 30 |
| | | * | Otherwise, let them go to operator | 40 |
| | | REDIRECT | QSECOFR | 50 |
| END | | | | 60 |

OPAL Example 5: Receive Save Files and Restore Objects

This example looks for a message that says that either file SLWSAVF or file PRODSAVF has arrived from another system. (The file and member names from the message are stored in OPAL variables FILE and MEMBER, respectively.) The example then clears the file of the same name, receives the file, restores the objects from the file, and sends a message back to the CALVIN system.

| Logic Operand | Variable | Operation | Operation Values | Seq |
|------------------|----------|-----------|-------------------------------------|-----|
| IF | FILE | EQ | SLWSAVF | 10 |
| | | EXECUTE | CLRSVF FILE(GGPL/FILE) | 20 |
| | | EXECUTE | RCVNETF FROMFILE(FILE) TOFILE(FI... | 30 |
| | | EXECUTE | RSTOBJ OBJ(*ALL) SAVLIB(SARA) DE... | 40 |
| | | EXECUTE | SNDNETMSG MSG('Cpps:e received S... | 50 |
| END | | | | 60 |
| IF | FILE | EQ | PRODSAVF | 70 |
| | | EXECUTE | CLRSVF FILE(GGPL/FILE) | 80 |
| | | EXECUTE | RCVNETF FROMFILE(FILE) TOFILE(QG... | 90 |
| | | EXECUTE | RSTOBJ OBJ(*ALL) SAVLIB(PRODLIB)... | 100 |
| | | EXECUTE | SNDNETMSG MSG('The production pr... | 110 |
| END | | | | 120 |

The full command for Seq 30 is as follows:

Extended Value RCVNETF FROMFILE(FILE) TOFILE(FILE) TOMBR(MEMBER)

The full command for Seq 40 is as follows:

Extended Value RSTOBJ OBJ(*ALL) SAVLIB(SARA) DEV(*SAVF) OBJTYPE(*FILE) SAVF(FILE)

The full command for Seq 50 is as follows:

Extended Value SENDNETMSG MSG('Cpps:e received SLW SAVF') TOUSRID((TERRI CALVIN))

The full command for Seq 90 is as follows:

Extended Value RCVNETF FROMFILE(FILE) TOFILE(QGPL/FILE) TOMBR(MEMBER)

The full command for Seq 100 is as follows:

Extended Value RSTOBJ OBJ(*ALL) SAVLIB(PRODLIB) DEV(*SAVF) SAVF(FILE)

The full command for Seq 110 is as follows:

Extended Value SENDNETMSG MSG('The production programs have been restored.') TOUSRID((TERRI CALVIN))

OPAL Example 6: TCP Job Ended

This OPAL code checks whether a message was received for job TCP. If so, it restarts the TCP job. If the message is for another job, the OPAL code suppresses it.

| Logic Operand | Variable | Operation | Operation Values | Seq |
|------------------|----------|-----------|-------------------------------------|-----|
| | | * | If TCP job ends, execute program to | 10 |
| | | * | restrat TCP job | 20 |
| IF | JOB | EQ | TCP | 30 |
| | | EXECUTE | SBMJOB CMD(CALL PGM(TCP)) | 40 |
| | | * | | 50 |
| | | * | Otherwise, suppress message | 60 |
| ELSE | | | | 70 |
| | | SUPPRESS | | 80 |
| END | | | | 90 |

OPAL Example 7: User Not Authorized to Change System Value

A message is sent if a user attempts to change a system value without authority to do so. If the user is QSYSOPR, a message is sent to JAKE. Otherwise, the message is sent to the user who tried to change the system value and the job that tried to change the system value is terminated.

| Logic Operand | Variable | Operation | Operation Values | Seq |
|------------------|----------|-----------|-------------------------------------|-----|
| | | * | Who tried to change the sysval? | 10 |
| | | * | Let the operator know - unless it | 20 |
| | | * | was him. Then, let Dick know. | 30 |
| IF | USER | EQ | QSYSOPR | 40 |
| | USRFLD1 | CHGTO | JAKE | 50 |
| ELSE | | | | 60 |
| | USRFLD1 | CHGTO | USER | 70 |
| | | EXECUTE | SNDMSG MSG('Someone is trying to... | 80 |
| | | * | | 90 |
| | | * | Kill their job | 100 |
| | | TERMINATE | | 110 |
| END | | | | 120 |

The full command for Seq 80 is as follows:

```
Extended Value  SNDMSG MSG('Someone is trying to change the system values. Th
eir job is being terminated.') TOUSR(USRFLD1)
```

OPAL Example 8: Cannot Allocate Library

When a message is received that a library cannot be allocated, the library name in the message is stored in OPAL variable LIBRARY. The OPAL code executes a WRKOBJLCK

command for the library to get a list of locks. It then sends the list to the host system and terminates the job that tried to allocate the library.

| Logic Operand | Variable | Operation | Operation Values | Seq |
|------------------|----------|-----------|-------------------------------------|-----|
| _____ | _____ | * | If someone has a library locked, | 10 |
| _____ | _____ | * | get a list of who has it locked | 20 |
| _____ | _____ | EXECUTE | WRKOBJLCK OBJ(QSYS/RBCMRGLIB) OB... | 30 |
| _____ | _____ | * | _____ | 40 |
| _____ | _____ | * | Send a report to the host | 50 |
| _____ | _____ | EXECUTE | SNDNETSPLF FILE(QPDSPOLK) TOUSRI... | 60 |
| _____ | _____ | * | _____ | 70 |
| _____ | _____ | * | Then, cancel the original program | 80 |
| _____ | _____ | TERMINATE | _____ | 90 |
| END | _____ | _____ | _____ | 100 |

The full command for Seq 30 is as follows:

Extended Value WRKOBJLCK OBJ(QSYS/RBCMRGLIB) OBJTYPE(*LIB)

The full command for Seq 60 is as follows:

Extended Value SNDNETSPLF FILE(QPDSPOLK) TOUSRID((TIM HOST))

OPAL Example 9: Line Failed—Probable Local Hardware Problem

When a line failed message is received, the line name is stored in the OPAL variable LINE. OPAL code then enters the reply **C** to cancel recovery procedures. It varies off the line, waits a minute, and varies on the line again.

| Logic Operand | Variable | Operation | Operation Values | Seq |
|------------------|----------|-----------|-------------------------------------|-----|
| _____ | _____ | * | Cancel recovery procedures and | 10 |
| _____ | _____ | * | vary line off | 20 |
| _____ | _____ | ENTER | C | 30 |
| _____ | _____ | EXECUTE | VRYCFG CFGOBJ(LINE) CFGTYPE(*LIN... | 40 |
| _____ | _____ | * | Wait 60 seconds for vary to finish | 50 |
| _____ | _____ | DELAY | 60 | 60 |
| _____ | _____ | * | Vary line back on | 70 |
| _____ | _____ | EXECUTE | VRYCFG CFGOBJ(LINE) CFGTYPE(*LIN... | 80 |
| END | _____ | _____ | _____ | 90 |

The full command for Seq 40 is as follows:

Extended Value VRFCFG CFGOBJ(LINE) CFGTYPE(*LIN) STATUS(*OFF)

The full command for Seq 80 is as follows:

Extended Value VRFCFG CFGOBJ(LINE) CFGTYPE(*LIN) STATUS(*ON)

OPAL Example 10: Produce a Dump

This example shows you how to produce a dump if a message hasn't been answered within 600 seconds. Robot Console answers the message with a D and then sends a message to user ROHIT stating that a dump has been produced.

| Logic Operand | Variable | Operation | Operation Values | Seq |
|------------------|----------|-----------|-------------------------------------|-----|
| _____ | _____ | RPYWITHIN | 600 | 10 |
| _____ | _____ | ENTER | D | 20 |
| _____ | _____ | EXECUTE | SNDMSG MSG('A dump has been prod... | 30 |
| END | _____ | _____ | _____ | 40 |

The full command for Seq 30 is as follows:

Extended Value SNDMSG MSG('A dump has been produced from FROMPGM') TOUSR(ROHIT)

OPAL Example 11: Answer Message and Remove from Message Center

If the message text contains the characters CRTRPG, the OPAL code answers the message with an S, then removes the message from the message center after it has been answered.

| Logic Operand | Variable | Operation | Operation Values | Seq |
|------------------|----------|-----------|------------------|-----|
| IF | TEXT | CT | CRTRPG | 10 |
| _____ | _____ | ENTER | S | 20 |
| _____ | _____ | REMOVE | _____ | 30 |
| END | _____ | _____ | _____ | 40 |

OPAL Example 12: Change Informational Message to Response Required

This example shows how to change an informational message to response required if the system time is between 1700 and 0700.

| Logic Operand | Variable | Operation | Operation Values | Seq |
|------------------|----------|-----------|------------------|-----|
| IF | SYSTIME | GE | 170000 | 10 |
| OR | SYSTIME | LE | 070000 | 20 |
| | | RESPOND | | 30 |
| END | | | | 40 |

OPAL Example 13: Use a Notification List

This example notifies users on the notification list SUPPORT if a message hasn't been answered within 300 seconds. If the message remains unanswered for another 600 seconds, Robot Console sends a message to the specified two-way device name. **Note:** The message sent by the PAGE2WAY operation includes the first-level message text. Depending on your system setup, it may also include the second-level text.

| Logic Operand | Variable | Operation | Operation Values | Seq |
|------------------|----------|-----------|------------------|-----|
| | | RPYWITHIN | 300 | 10 |
| | | NOTIFYL | SUPPORT | 20 |
| | | RPYWITHIN | 600 | 30 |
| | | PAGE2WAY | DD2WAY | 40 |
| END | | | | 50 |

OPAL Example 14: Use RBTBCHUPD to Run a Robot Schedule Job

This example checks for the characters PREDIT in a message and then checks the system time. If the time is less than 1800, it delays the message and then uses the RBTBCHUPD command to run the Robot Schedule job 000000000011; otherwise it runs the Robot Schedule job.

| Logic Operand | Variable | Operation | Operation Values | Seq |
|------------------|----------|-----------|-------------------------------------|-----|
| IF | DATA | CT | PREDIT | 10 |
| IF | SYSTIME | LT | 180000 | 20 |
| | | EXECUTE | DLYJOB RSMTIME(180000) | 30 |
| | | EXECUTE | ROBOTLIB/RBTBCHUPD JOBNUMBER(000... | 40 |
| ELSE | | | | 50 |
| | | EXECUTE | ROBOTLIB/RBTBCHUPD JOBNUMBER(000... | 60 |
| END | | | | 70 |
| END | | | | 80 |

For the Seq 40 and 60 lines, type the first part of the command **ROBOTLIB/RBTBCHUPD** and press function key 4 to open the command screen where you can enter the parameters.

Note: In Robot Console Explorer, all the parameters are displayed in the window. On the IBM i, press **Page Down** to see the rest of the parameters.

Update ROBOT in Batch (RBTBCHUPD)

Type choices, press Enter.

| | | | |
|---------------------------------|-------------------|-----------------|--|
| ROBOT Job Name | | Name | |
| ROBOT Job Number | > 000000000011 | | |
| Job Type | - | W, P, C, G | |
| Application | | Character value | |
| Description | | | |
| Time(s) to run this job | | 0001-2357 | |
| | + for more values | | |
| Run on Monday | - | Y, N, L, 1-5 | |
| Tuesday | - | Y, N, L, 1-5 | |
| Wednesday | - | Y, N, L, 1-5 | |
| Thursday | - | Y, N, L, 1-5 | |
| Friday | - | Y, N, L, 1-5 | |
| Saturday | - | Y, N, L, 1-5 | |
| Sunday | - | Y, N, L, 1-5 | |
| Dates to run this job | | Character value | |
| | + for more values | | |

More...

F3=Exit F4=Prompt F5=Refresh F12=Cancel F13=How to use this display
F24=More keys

Type choices, press Enter.

| | | | |
|------------------------------|-------------------|-----------------------------|--|
| Schedule Override | D | D, E, H, HA, N, O, R, RA... | |
| D0 Special Options | | *OPAL, *NOREACT, *CLEAR | |
| | + for more values | | |

OPAL Example 15: Check OPAL Table for Variable Value

This example checks the OPAL table, SARATBL, to see if the value in OPAL variable VAR4 is contained in the table. If it is, the message is changed to response required; if not, the message is answered with the default reply.

| Logic | Operand | Variable | Operation | Operation Values | Seq |
|-------|---------|----------|-----------|------------------|-----|
| IF | VAR4 | INTABLE | SARATBL | | 10 |
| | | RESPOND | | | 20 |
| ELSE | | | | | 30 |
| | | DEFAULT | | | 40 |
| END | | | | | 50 |

OPAL Example 16: Send a System Alert and Break Message

This example sends a system alert to user SARA and then sends a break message to her workstation.

| Logic Operand | Variable | Operation | Operation Values | Seq |
|------------------|----------|-----------|-------------------------------------|-----|
| | | SNDALERT | SARA | 10 |
| | | EXECUTE | SNDBRKMSG MSG('An alert has been... | 20 |
| END | | | | 30 |

The full command for Seq 20 is as follows:

```
Extended Value SNDBRKMSG MSG('An alert has been sent for the following messa
ge ID: CPF2141.') TOMSGQ(SARAS1)
```

OPAL Example 17: Change Message to Response Required

This example checks if the job that created an error is RBCNOTIFY or if the job name contains the characters MS. If it does, the message set stops processing. Otherwise, the message is changed to response required.

| Logic Operand | Variable | Operation | Operation Values | Seq |
|------------------|----------|-----------|------------------|-----|
| IF | JOB | EQ | RBCNOTIFY | 10 |
| OR | JOB | CT | MS | 20 |
| | | QUIT | | 30 |
| ELSE | | | | 40 |
| | | RESPOND | | 50 |
| END | | | | 60 |

OPAL Example 18: Answer Message Depending on Device Name

This example checks if the device in the message is PRT01. If it isn't, it answers the message with a G; if it is, the message is answered with a 0.

| Logic Operand | Variable | Operation | Operation Values | Seq |
|------------------|----------|-----------|------------------|-----|
| IF | DEVICE | NE | PRT01 | 10 |
| | | ENTER | G | 20 |
| ELSE | | | | 30 |
| | | ENTER | 0 | 40 |
| END | | | | 50 |

OPAL Example 19: Send Message or Reset Device

This example checks if the system time is between 1800 and 0700. If it is, it changes USRFLG1 to Y. It also checks if the device name contains the characters ASYNC. If it does, it changes USRFLG1 to Y. If USRFLG1 is equal to Y, it sends a message to the device DD2WAY; otherwise, it resets the device. **Note:** The message sent by the PAGE2WAY operation includes the first-level message text. Depending on your system setup, it may also include the second-level text.

| Logic Operand | Variable | Operation | Operation Values | Seq |
|------------------|----------|-----------|-------------------------------------|-----|
| IF | SYSTIME | GT | 180000 | 10 |
| OR | SYSTIME | LT | 070000 | 20 |
| THEN | USRFLG1 | CHGTO | Y | 30 |
| END | | | | 40 |
| IF | DEVICE | CT | ASYNC | 50 |
| | USRFLG1 | CHGTO | Y | 60 |
| END | | | | 70 |
| IF | USRFLG1 | EQ | Y | 80 |
| | | PAGE2WAY | DD2WAY | 90 |
| ELSE | | | | 100 |
| | | EXECUTE | VRYCFG CFGOBJ(DEVICE) CFGTYPE(*D... | 110 |
| END | | | | 120 |

The full command for Seq 110 is as follows:

Extended Value VRYCFG CFGOBJ(DEVICE) CFGTYPE(*DEV) STATUS(*ON) RESET(*YES)

OPAL Example 20: Suppress a Message or Send a Message

This example checks if the controller name in the message text contains the characters PC. If it does, it suppresses the message; otherwise it sends a message to a device or broadcast list. **Note:** The message sent by the PAGE2WAY operation includes the first-level message text. Depending on your system setup, it may also include the second-level text.

| Logic Operand | Variable | Operation | Operation Values | Seq |
|------------------|------------|-----------|------------------|-----|
| IF | CONTROLLER | CT | PC | 10 |
| | | SUPPRESS | | 20 |
| ELSE | | | | 30 |
| | | PAGE2WAY | DD2WAY | 40 |
| END | | | | 50 |

OPAL Example 21: End a Job

This example ends a job that produced a message.

| Logic | Operand | Variable | Operation | Operation Values | Seq |
|-------|---------|----------|-----------|------------------|-----|
| | | | TERMINATE | | 10 |
| | END | | | | 20 |

OPAL Example 22: Answer a Repeated Message

This example checks if a message has been repeated more than 5 times. If it has, it redirects the message to user TERRI and waits 300 seconds for a reply. If the message remains unanswered, it answers the message with a **C**. If the message has been repeated fewer than 5 times, it answers the message with an **I**.

| Logic | Operand | Variable | Operation | Operation Values | Seq |
|-------|---------|----------|-----------|------------------|-----|
| | IF | REPEAT | GT | 5 | 10 |
| | | | REDIRECT | TERRI | 20 |
| | | | RPYWITHIN | 300 | 30 |
| | | | ENTER | C | 40 |
| | ELSE | | | | 50 |
| | | | ENTER | I | 60 |
| | END | | | | 70 |

OPAL Example 23: Redirect Messages to an Active User

This example checks if user TERRI is signed on and using Robot Console. If she is, it redirects messages to her.

| Logic | Operand | Variable | Operation | Operation Values | Seq |
|-------|---------|----------|-----------|------------------|-----|
| | IF | ACTUSR | EQ | TERRI | 10 |
| | | | REDIRECT | TERRI | 20 |
| | END | | | | 30 |

OPAL Example 24: Redirect Messages to HOSTCENTER

This example checks if QSYSOPR is signed on. If not, it redirects message to HOSTCENTER.

Note: You must have Robot Network installed.

| Logic | Operand | Variable | Operation | Operation Values | Seq |
|-------|---------|----------|-----------|------------------|-----|
| | IF | ACTUSR | NE | QSYSOPR | 10 |
| | | | REDIRECT | HOSTCENTER | 20 |
| | END | | | | 30 |

Quick Reference Guide

Robot Console Quick Reference Guide

Logic Operands

| | | | |
|-------|-----------|------------|--------|
| IF | variable | comparison | value |
| WHILE | variable | comparison | value |
| AND | variable | comparison | value |
| OR | variable | comparison | value |
| THEN | variable* | operation | value* |
| ELSE | — | — | — |
| | variable* | operation | value* |
| END | — | — | — |

* optional

Comparisons

| | |
|-------------|--------------------------|
| EQ | Equal |
| CT | Contains |
| DC | Does not contain |
| GT | Greater than |
| GE | Greater than or equal to |
| INMSGTBL | In message table |
| INTABLE | In OPAL table |
| LT | Less than |
| LE | Less than or equal to |
| NE | Not equal |
| NOTINMSGTBL | Not in message table |
| NOTINTABLE | Not in OPAL table |

Operations

| Operation | Operation Value |
|------------|-----------------------------------|
| ADD | Amount to increment |
| BCAT | Value to concatenate (with blank) |
| CALLCP | Program to process message |
| CALLCP3 | Program to process V3 message |
| CALLRPY | Program to process reply |
| CAT | Value to concatenate (no blank) |
| CHGTO | New value |
| COPY | Message center |
| DEFAULT | None (enters default reply) |
| DELAY | Number of seconds |
| ENTER | Reply |
| EXECUTE | F4 (enter command) |
| GOTO | Tag statement |
| NOTIFY | User to notify |
| NOTIFYL | Notification list name |
| OPERATOR | None (stop and wait for reply) |
| PAGE | F4 (enter message) |
| PAGE2WAY | Two-way device name |
| QUIT | None (ends message processing) |
| REDIRECT | Message center name |
| REMOVE | None |
| RESPOND | None (response required) |
| RPYWITHIN | Time limit in seconds |
| SENDMC | F4 (enter new message) |
| SENDQ | F4 (enter message queue) |
| SNDAAlert | Resource name |
| SNDRBTDTA | F4 (enter Robot Schedule prereq.) |
| SNDSNMPMSG | None |

| Operation | Operation Value |
|-----------|-----------------------------------|
| SNDSNMPPY | None |
| SUB | Value to subtract |
| SUPPRESS | None |
| TAG | Tag name |
| TERMINATE | None (ends job that sent message) |
| * | Comment text |

Variables

| OPAL Variable | Value |
|---------------|----------------------------|
| ACTCTL | Is this controller active? |
| ACTDEV | Is this device active? |
| ACTLIN | Is this line active? |
| ACTUSR | Is this user active? |
| CALENDAR | Robot Schedule calendar |
| CENTER | Message center |
| CONTROLLER | Controller name |
| DATA | Message data |
| DAY | Day of week (1-7, Mon=1) |
| DAYMTH | Day number in month |
| DEVICE | Device name |
| DISTRIBQ | Distribution queue name |
| FILE | File name |
| FILTERNAME | Filter triggering event |
| FORMTYPE | Form type name |
| FROMPGM | Program that sent message |
| JOB | Job that sent message |
| JOBNUMBER | Job number |
| JOURNAL | Journal name |

| OPAL Variable | Value |
|-----------------------|---|
| LASTDAY | Last date in month |
| LIBRARY | Library name |
| LINE | Line name |
| LOGDATA | Log data being monitored |
| MEMBER | Member name |
| MSGDATE | Message date |
| MSGGRP | Message group |
| MSGID | Message identifier |
| MSGSEV | Message severity (0-99) |
| MSGSTS | Message status |
| MSGTIME | Time message was sent |
| MSGTYPE | Message type (01-25, LI, LQ) |
| NO | False comparison to USRFLG _n |
| NUMBER | Network file number |
| OBJTYP | Extended resource object type |
| PRIORMRS _n | Earlier message group/set ($n=1-5$) |
| PRIORMSG _n | Earlier message ID ($n=1-5$) |
| PRIORRPY _n | Earlier reply ($n=1-5$) |
| PROGRAM | Program name |
| RCVNBR | Receiver job number |
| RCVUSR | Receiver user |
| RECEIVER | Receiver job name |
| REPEAT | Count (match msg ID and data) |
| REPEATJOB | Count (match message job) |
| REPEATMSG | Count (match message ID) |
| RESOURCE | Resource name |
| ROUTER | Router job name |
| ROUTERNBR | Router job number |
| ROUTERUSR | Router user |

| OPAL Variable | Value |
|---------------|--|
| RPYCENTER | Message center that replied |
| RPYJOB | Job that replied |
| RPYJOBNBR | Job number that replied |
| RPYTEXT | Reply text |
| RPYUSER | User who replied |
| RSCID | Resource ID |
| SENDER | Sender job name |
| SENDERNBR | Sender job number |
| SENDERUSR | Sender user |
| STATUS | Resource status |
| STRING nn | String from log ($nn=01-18$) |
| SUBSYSTEM | Subsystem name |
| SYSDATE | Current system date |
| SYSTEM | System name |
| SYSTIME | Current system time |
| TEXT | First-level message text |
| TXT2NDLVL | Second-level message text |
| TYPE | Resource type |
| USRTEXT | Custom message text (512 max) |
| USRTEXT2 | Custom message text (1000 max) |
| USER | Job user |
| USRFLD n | 10-character variables ($n=1-5$) |
| USRFLAG n | 1-character flag variables ($n=1-5$) |
| USRNBR n | Numeric variables (5,0) ($n=1-5$) |
| VAR n | Message variables ($n=1-30$) |
| WEEKNO | Week number in month |
| WORKDAY | Is this a working day? |
| WRITER | Writer name |
| YES | True comparison to USRFLG n |