

Report Modernization Utility for z/OS and OS/390



# User's Guide and Reference

*Version 1 Release 1*



Report Modernization Utility for z/OS and OS/390



# User's Guide and Reference

*Version 1 Release 1*

**Note!**

Before using this information and the product it supports, be sure to read the general information under "Notices" on page 65.

**First Edition (March 2009)**

This edition applies to Report Modernization Utility for z/OS Version 1 Release 1, Program Number 5697-N44 and to any subsequent releases until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

The information in this manual was furnished by Foundation Software, Inc.

This publication is available on the Web at:

<http://www.ibm.com/software/awdtools/migration>

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

A form for readers' comments is provided at the back of this publication. If the form has been removed, address your comments to:

IBM Corporation  
H150/090  
555 Bailey Avenue  
San Jose, CA  
95141-1003  
U.S.A.

or fax your comments from within the U.S.A., to 800-426-7773, or, from outside the U.S.A., to 408-463-2629.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright Foundation Software, Inc. 2009.

All rights reserved. Unauthorized use or disclosure of any part of the system is prohibited. Foundation Software, Inc. has granted IBM a non-exclusive license to market RMU as Report Modernization Utility.

© Copyright International Business Machines Corporation 2009.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

---

# Contents

<b>About this manual</b> . . . . .	<b>v</b>	Decorating an HTML document . . . . .	29
<b>Who should use this manual</b> . . . . .	<b>vii</b>	Debugging RMU Script programs . . . . .	30
<b>Structure of this manual</b> . . . . .	<b>ix</b>	<b>Chapter 5. RMU Script language</b>	
<b>How to read syntax diagrams</b> . . . . .	<b>xi</b>	<b>instruction reference.</b> . . . . .	<b>33</b>
<b>Chapter 1. Introducing RMU</b> . . . . .	<b>1</b>	PARM statement. . . . .	33
How does it work? . . . . .	1	DEFINE statement . . . . .	34
RMU features . . . . .	1	<object> and </object> tag . . . . .	35
<b>Chapter 2. Working with RMU</b> . . . . .	<b>3</b>	<style> and </style> tag . . . . .	35
Running RMU. . . . .	3	<docs_top> and </docs_top> tag . . . . .	36
Distributing output documents . . . . .	4	<docs_end> and </docs_end> tag. . . . .	37
Summary of available jobs. . . . .	5	<page_top> and </page_top> tag . . . . .	37
Summary of file DD Names . . . . .	5	<page_end> and </page_end> tag . . . . .	38
File descriptions . . . . .	7	Assignment statement. . . . .	38
Using the FZHBPARS parser utility. . . . .	8	BYPASS statement . . . . .	41
<b>Chapter 3. Installation and RMU</b>		CALL statement . . . . .	42
<b>options</b> . . . . .	<b>11</b>	CONTINUE statement. . . . .	42
Installation . . . . .	11	CONTROL statement . . . . .	43
RMU default options (FZHOPTAB) table . . . . .	11	DISPLAY statement. . . . .	44
RMU default library locator table (FZHPROCS)	13	DO and END-DO statements . . . . .	44
Setting up the z/OS server (UNIX) environment	14	EVALUATE and END-EVALUATE statements . . . . .	45
System information. . . . .	15	IF, ELSE, and END-IF statements . . . . .	47
<b>Chapter 4. Working with RMU Script</b> . . . . .	<b>17</b>	PERFORM statement . . . . .	47
Basic concepts . . . . .	17	PROC/END-PROC statement . . . . .	48
I/O handling. . . . .	17	PROCESS statement . . . . .	49
Coding rules . . . . .	18	STOP statement . . . . .	49
Program example . . . . .	19	STRING statement . . . . .	50
Creating CSV documents. . . . .	21	System-defined fields . . . . .	51
Creating simple HTML documents . . . . .	24	COBOL verbs, statements, and reserved fields. . . . .	52
Creating HTML1 documents with the control breaks			
selection tree . . . . .	26		
		<b>Chapter 6. Messages.</b> . . . . .	<b>59</b>
		RMU compiler-generated messages . . . . .	59
		<b>Notices</b> . . . . .	<b>65</b>
		Trademarks . . . . .	66
		<b>Index</b> . . . . .	<b>67</b>



---

## About this manual

This manual describes how to use the Report Modernization Utility for z/OS licensed program.

In the rest of this document, the Report Modernization Utility for z/OS is referred to as "RMU".





---

## Who should use this manual

This manual is for anyone who wants to convert standard mainframe reports to HTML or Character Separated Values (CSV) file format.

To use RMU properly, you need to be familiar with:

- Job Control (JCL)
- The RMU script language (RMU Script)
- COBOL compiling procedures



---

## Structure of this manual

This manual consists of:

- Chapter 1, “Introducing RMU,” on page 1.  
Explains what RMU is and how it works.
- Chapter 2, “Working with RMU,” on page 3.  
An overview of the supplied JCL.
- Chapter 3, “Installation and RMU options,” on page 11.  
Describes RMU installation and customization.
- Chapter 4, “Working with RMU Script,” on page 17.  
Describes in detail how to work with the RMU Script language.
- Chapter 5, “RMU Script language instruction reference,” on page 33.  
Describes statements supported by RMU.
- Chapter 6, “Messages,” on page 59.  
Provides a list of:
  - COBOL compiler generated messages
  - RMU runtime error messages
  - RMU compiler generated messages



## How to read syntax diagrams

The syntactical structure of commands described in this document is shown by means of syntax diagrams.

Figure 1 shows a sample syntax diagram that includes the various notations used to indicate such things as whether:

- An item is a keyword or a variable.
- An item is required or optional.
- A choice is available.
- A default applies if you do not specify a value.
- You can repeat an item.

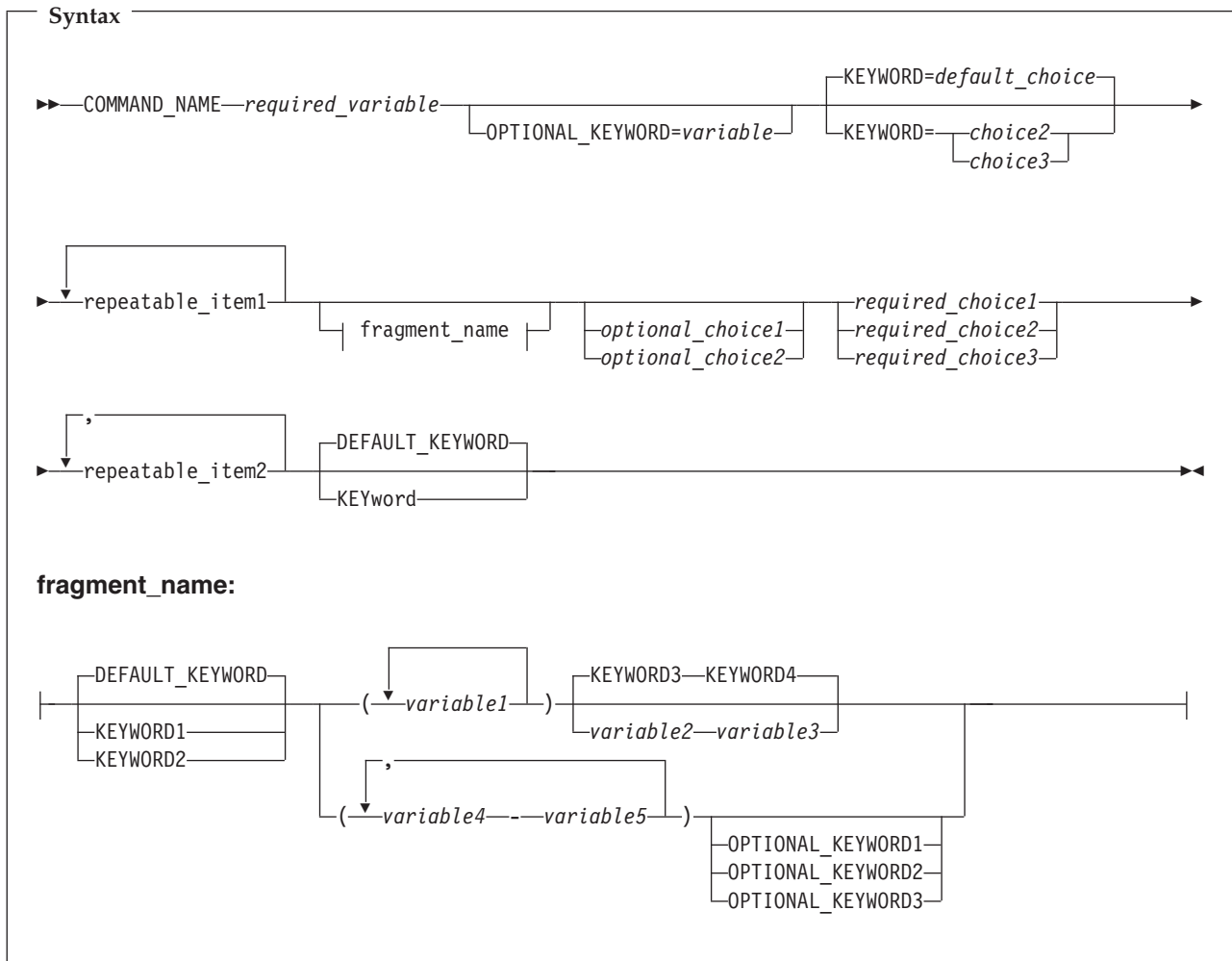


Figure 1. Sample syntax diagram

Here are some tips for reading and understanding syntax diagrams:

**Order of reading**

Read the syntax diagrams from left to right, from top to bottom, following the path of the line.

The ► symbol indicates the beginning of a statement.

The → symbol indicates that a statement is continued on the next line.

The ◀ symbol indicates that a statement is continued from the previous line.

The →◀ symbol indicates the end of a statement.

**Keywords**

Keywords appear in uppercase letters.



Sometimes you only need to type the first few letters of a keyword. The required part of the keyword appears in uppercase letters.



In this example, you could type "KEY", "KEYW", "KEYWO", "KEYWOR" or "KEYWORD".

The abbreviated or whole keyword you enter must be spelled exactly as shown.

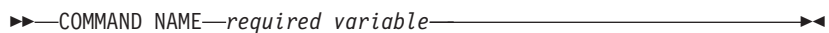
**Variables**

Variables appear in lowercase letters. They represent user-supplied names or values.



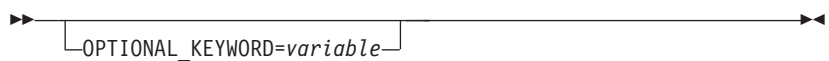
**Required items**

Required items appear on the horizontal line (the main path).



**Optional items**

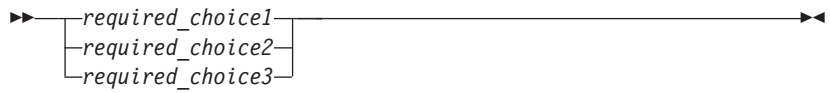
Optional items appear below the main path.



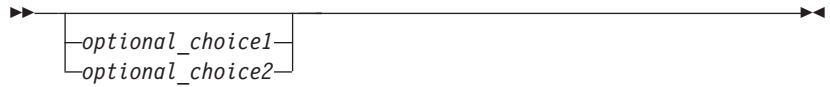
**Choice of items**

If you can choose from two or more items, they appear vertically, in a stack.

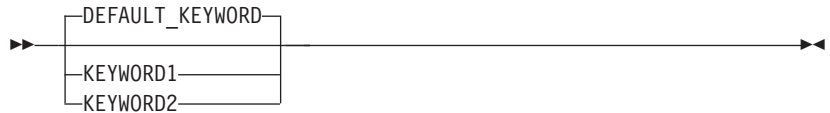
If you *must* choose one of the items, one item of the stack appears on the main path.



If choosing one of the items is optional, the entire stack appears below the main path.



If a default value applies when you do not choose any of the items, the default value appears above the main path.



### Repeatable items

An arrow returning to the left above the main line indicates an item that can be repeated.



If you need to specify a separator character (such as a comma) between repeatable items, the line with the arrow returning to the left shows the separator character you must specify.



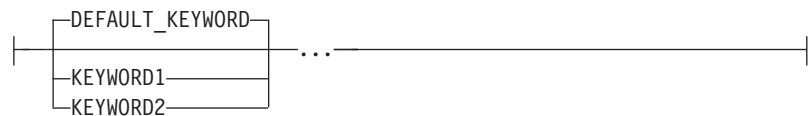
### Fragments

Where it makes the syntax diagram easier to read, a section or *fragment* of the syntax is sometimes shown separately.



⋮

**fragment\_name:**







---

## Chapter 1. Introducing RMU

RMU is a stand-alone utility that converts mainframe printer files (reports) to HTML and CSV files. It runs exclusively on the z/OS<sup>®</sup> platform.

The purpose of RMU is to provide a comprehensive tool for converting mainframe reports to an HTML and CSV format which is understood by standard browsers and spreadsheet software such as IE and EXCEL, without the need to modify existing programs. Once the reports are converted, they can be published to z/OS or any other server, or downloaded to a PC platform for local access.

---

### How does it work?

RMU executes on z/OS in background as a batch job. The input and output environment is controlled by means of JCL. RMU Script is available for massaging and decorating the input. The output is published directly to the z/OS server, or written to a file that can be routed or downloaded to any other server or client platform.

---

### RMU features

The main features within RMU are:

- HTML and CSV documents can be produced from standard reports without changing the application programs.
- RMU Script is available for decorating HTML documents. The language can:
  - Decorate HTML text by means of fonts, CSS, colors, and so on. For example:
    - Negative amounts can be turned red.
    - The background can simulate 1403 green-striped printer paper.
  - Place objects such as pictures and special annotations at specific positions in the document.
  - Be used in interpretive mode or compiled and linked mode.
- Simple HTML documents can be produced (the use of RMU Script is optional).
- HTML documents, with a control breaks tree on the left side and the report body on the right side, can be produced (the use of RMU Script is required).
- HTML documents can be published to the z/OS server by pointing the JCL to the z/OS server.
- HTML documents can be created to a disk file and downloaded to any other platform.
- CSV files can be produced (the use of RMU Script is required).

## RMU features

---

## Chapter 2. Working with RMU

This section provides an overview of the JCL supplied with RMU.

---

### Running RMU

RMU can be run as a stand-alone batch utility in three ways:

#### Running RMU without RMU Script

This is the simplest form. The FZHRMU00 program is run without the SYSIN for RMU Script. The input report is read and an output HTML-format document is created without custom decorations. The background color and turf is supplied by the PARM= ('STYLE=&css,TURF=&turf') options on the EXEC statement.

```
//RUNRMU EXEC PGM=FZHRMU00,PARM=('STYLE=&css,TURF=&turf')
```

Where:

**&css** PDS member containing CSS in FJIDOC0 PDS. This is an optional parameter.

**&turf** The background color. Valid values are:

**1403-paper** Simulate 1403 green-striped printer paper. This is the default.

**NO** Create plain solid background color.

Use the supplied FZHRMUJ0 or FZHRMUX0 job located in the &SYS1.SFZHJCLS library as a template for your own jobs.

#### Running RMU with compiled RMU Script

In this mode, the FZHRMU00 program is run by invoking a compiled RMU Script program. The name of the RMU Script program to use and the style CSS are supplied by the PARM= on the EXEC statement.

```
//RUNCOMP EXEC PGM=FZHRMU00,PARM=('SCRIPT=&pgmname,STYLE=&css')
```

Where:

**&pgmname** The name of the script program. This is a required parameter.

**&css** The PDS member containing CSS in FJIDOC0 PDS. Style is used only if there is no style method coded in the RMU Script. This is an optional parameter.

The SYSIN for the Script source must not be coded in the JCL.

The **&pgmname**, RMU SCRIPT, must have been previously compiled and linked by means of the FZHRMU01 compiler.

Use the supplied FZHLINKJ job located in the &SYS1.SFZHJCLS library as a template to compile and link RMU Script programs.

Use the supplied FZHRMUJ1 or FZHRMUX1 job located in the &SYS1.SFZHJCLS library as a template to run the compiled RMU Script program.

## Running RMU

### Running RMU with RMU Script in "link and go" mode

In this mode, the FZHRMU00 program compiles and links RMU Script pointed to by the SYSIN and then runs it. SYSIN can be a flat file, a PDS or PDSE member, or an inline source. The background style can be supplied by the PARM= on the EXEC statement.

```
//RUNCOMP EXEC PGM=FZHRMU00,PARM=(STYLE=&css)
```

Where:

**&css** PDS member containing CSS in FJIDOC0 PDS. Style is used if there is no style method coded in the RMU Script. This is an optional parameter.

The SYSIN is required.

"PARM LINK (&NAME)" must not be specified on the PARM statement in the RMU Script program.

Use the supplied FZHRMUJ2 or FZHRMUX2 job located in the &SYS1.SFZHJCLS library as a template to run as "link and go".

---

## Distributing output documents

RMU can publish the output documents to the z/OS server (UNIX®), or to a regular z/OS flat variable-length file.

When documents are published to a regular z/OS variable-length file, the file must be downloaded to a PC (or another server) for access.

- **When you create simple HTML format**, download the output file as an ASCII text file with an extension of .htm. The downloaded file can be opened with IE or any other compatible browser.
- **When you create HTML1 format**, download the output as a binary file and run the FZHBPARS Java™ parser to build the required directories. See "Using the FZHBPARS parser utility" on page 8 for detailed steps.
- **When you create a CSV file**, download the output file as an ASCII text file with an extension of .csv. The downloaded file then can be imported into any spreadsheet software.

When documents are published to the z/OS server, the z/OS UNIX environment must be established and enabled. A root directory on the z/OS UNIX system must be established for each user. Coordinate access to the UNIX environment with your z/OS system administrator.

RMU determines where to send the output by checking for the existence of the name FJUNIX0 DD in the JCL. PATH= on the FJUNIX0 definition must point to the root directory showing where to write the documents.

If the name FJUNIX0 DD exists in the JCL, RMU assumes that the HTML documents are being sent to a z/OS UNIX System. If the name FJUNIX0 DD does not exist, RMU creates a flat file on z/OS.

**Note:** UNIX files are handled by the FZHUNIX1 RMU program. This program is dynamically loaded at end of job when combining the output. UNIX is case-sensitive. That is, commands, directory, and file names must be typed exactly as defined on the UNIX system.

When publishing to the z/OS server, RMU must comply with the code set (ASCII or EBCDIC) defined for each document type on the UNIX system. The code set is obtained from the UNIX file, `httpd.conf`.

**Note:** The layout of the UNIX file, `httpd.conf`, is beyond the scope of this document. Consult with your z/OS system administrator for the location of this file.

The `httpd.conf` file is pointed to by the `FJCONFIG` DD name in the JCL. It can be a z/OS PDS member or a file in the UNIX directory.

If the `FJCONFIG` DD name is not in the JCL, ASCII is assumed for HTML and CSV files, and binary code for gif files.

For a list of available job templates in the `&SYS1.SFZHJCLS` library, see “Summary of available jobs.”

For a list of files needed in the JCL, see “Summary of file DD Names.”

---

### Summary of available jobs

These jobs are available:

RMU Script compiler job (FZHRMU01 program):

**FZHLINKJ**      Compile and link RMU Script program.

Jobs to use when the output documents are to be kept on z/OS for downloading to a PC (or other non-z/OS server):

**FZHRMUJ0**      Run RMU without the RMU Script program.

**FZHRMUJ1**      Run the compiled RMU Script program.

**FZHRMUJ2**      Run RMU Script as a “link and go” job.

Jobs to be used when output documents are to be distributed to the z/OS server (UNIX on z/OS):

**FZHRMUX0**      Run RMU without the RMU Script program.

**FZHRMUX1**      Run the compiled RMU Script program.

**FZHRMUX2**      Run RMU Script as a “link and go” job.

---

### Summary of file DD Names

These files are required when running RMU without the RMU Script program:

*Table 1. Files required when running RMU without the RMU Script*

File	Required or optional
FJIRPT0	Required
FJORPT0	Required
FJIDOC0	Optional
SYSOUT	Required

These files are required when compiling RMU Script:

*Table 2. Files required when when compiling RMU Script*

File	Required or optional
COBLIST	Optional

## Summary of file DD Names

*Table 2. Files required when when compiling RMU Script (continued)*

<b>File</b>	<b>Required or optional</b>
FJSVC99	Optional
LKEDMAP	Optional
RMULIST	Optional
RMUERR1	Optional
SYSLMOD	Required
SYSIN	Required

These files are required when running RMU with compiled RMU Script:

*Table 3. Files required when running RMU with compiled RMU script*

<b>File</b>	<b>Required or optional</b>
FJIRPT0	Required
FJORPT0	Required
FJIDOC0	Optional
FJSVC99	Optional
SYSOUT	Required

These files are required when running RMU in "link and go" mode:

*Table 4. Files required when running RMU in "link and go" mode*

<b>File</b>	<b>Required or optional</b>
COBLIST	Optional
FJIRPT0	Required
FJORPT0	Required
FJIDOC0	Optional
FJSVC99	Optional
LKEDMAP	Optional
RMULIST	Optional
RMUERR1	Optional
SYSOUT	Required
SYSIN	Required

These are additional files needed when publishing to z/OS UNIX:

*Table 5. Additional files needed when publishing to z/OS UNIX*

<b>File</b>	<b>Required or optional</b>
FJUNIX0	Required
FJDMAP0	Optional
STDERR	Optional
STDOUT	Optional
FJCONFIG	Optional

## File descriptions

### COBLIST

Output COBOL compiler listing.

RECFM=FBA, LRECL=133.

This file is created when compiling RMU Script and the COBOL option is coded on the PARM in the RMU Script program. This is an optional file. The default is SYSOUT=\*

### FJCONFG

Optional z/OS UNIX configuration file for determining the code set for file types. The file is used when publishing the document to the z/OS UNIX server. It can be a z/OS PDS member, or a file in the UNIX directory. If FJCONFG is not in the JCL, ASCII is assumed for HTML and CSV files, and binary code for gif files.

### FJDMAP0

Output list of files published to the z/OS server (z/OS UNIX).

RECFM=VB, LRECL=4096

This is an optional file. The default is SYSOUT=\*

### FJIDOC0

Input file where HTML templates and CSS files are located.

RECFM=VB, LRECL=4096

This is an optional file. The default is &SYS1.SFZHDOCS PDS. If you are creating custom CSS files, place the FJIDOC0 DD name in the JCL. Concatenate &SYS1.SFZHDOCS first, followed by your own data set names.

### FJIRPT0

The input report (report to be converted).

This can be an FBA/VBA or an FB/VB file. Maximum LRECL=512 (including cc).

The first byte must be a valid ANSI print control character.

### FJORPT0

The output report (HTML or CSV document).

RECFM=VB, LRECL=4096

When distributing to z/OS UNIX, this file is used as a work file. When creating a z/OS flat file, this file must be downloaded to a PC (or another server). The HTML1 format output must be parsed with the FZHBPARS parser utility. For details, see "Using the FZHBPARS parser utility" on page 8.

### FJSVC99

This is an optional DD name for turning SVC99 messages ON or OFF. This is a dummy temporary file used to control RMU's dynamic allocator messages. If supplied, it must be coded as follows:

```
//FJSVC99 DD DSN=&&FJSVC99(&svc99),DISP=NEW
```

Where:

*&svc99*

Can have one of the values: MSGOFF, MSGALL, or MSGTXT.

## Summary of file DD Names

### FJUNIX0

z/OS UNIX path (directory on z/OS UNIX showing where to publish the output). This is an HFS file. The maximum buffer length is 4096. The specified path must exist on z/OS UNIX with read and write permissions. When publishing to z/OS UNIX, RMU uses this path as the base directory into which all other created subdirectories and files are placed.

### FJUNIX1

z/OS UNIX HFS file internally allocated by RMU. The files are dynamically allocated by RMU. Do not code this file in the JCL.

### LKEDMAP

Link Edit map from the link step.

RECFM=FBA, LRECL=133

This file is created when compiling RMU Script and the LKED option is coded on the PARM in the RMU Script program. This is an optional file. The default is SYSOUT=\*

### RMULIST

RMU Script compiler listing.

RECFM=FBA, LRECL=133

This file is produced by the RMU Script compiler. This is an optional file. The default is SYSOUT=\*

### RMUERR1

RMU Script compiler errors.

RECFM=FBA, LRECL=133

This file is produced by the RMU Script compiler. This is an optional file. The default is SYSOUT=\*

### STDERR

Standard z/OS UNIX error file. When publishing to z/OS UNIX, potential UNIX errors are written to this file. The default is STDERR in the home directory on z/OS UNIX.

### STDOUT

Standard z/OS UNIX messages file. When publishing to z/OS UNIX, standard UNIX messages are written to this file. The default is STDOUT in the home directory on z/OS UNIX.

### SYSIN

SYSIN of the RMU Script program.

RECFM=F, LRECL=80

This is an optional file. If present, the use of the RMU script compiler is assumed.

### SYSOUT

Standard SYSOUT file for runtime messages required by every job.

---

## Using the FZHBPARS parser utility

For the HTML1 option, RMU generates a binary format file that contains several HTML, gif, and CSS documents. The downloaded file must be parsed with the Java program, FZHBPARS, before it can be viewed with a browser.

To install the parser program FZHBPARS:



1. On your PC or a server, create a directory for the RMU documents.  
This is an example of a Windows® command line to create a directory called RMUDOCs:  

```
md rmudocs
```
2. Download from the mainframe (in binary format with no LF/CR) &SYS1.SFZHDOCS(fzhbpars) into the created directory:  

```
c:\rmudocs\fzhbpars.class
```

To view an HTML1 document on your PC or a server, perform these tasks:

1. Make sure that Java VM is installed and running.
2. Download the HTML1 document file from the mainframe (in binary format, no LF/CR) into a directory (assume the rmudocs directory).
3. Switch to the rmudocs directory:  

```
cd\rmudocs
```
4. Enter on the command line:  

```
java fzhbpars &src &dest
```

Where:

**&src** The downloaded HTML1 document file.

**&dest** The target directory for the HTML1 documents.

For example (assuming the file name htmlfil1.bin):

```
java fzhbpars htmlfil1.bin htmlfil1
```

At completion, all HTML documents are created in the c:\rmudocs\htmlfil1\r001 directory (drive c: assumed). To view from your default browser, click on the c:\rmudocs\htmlfil1\r001\index.htm file. Alternatively, you can open your browser and enter in the URL:

```
file:///C:/rmudocs/htmlfil1/r001/index.htm
```

When you click on index.htm, a selection tree on the left and the report body on the right side of the screen is displayed. When you click on FJORPT0.htm, the report body is displayed on the screen.



---

## Chapter 3. Installation and RMU options

This section describes RMU installation and customization.

---

### Installation

RMU is installed using SMP/E. Refer to the *Program Directory* (GI10-8807-00) for installation instructions.

RMU consists of these six libraries:

<b>&amp;SYS1.SFZHAMAC</b>	Assembler (BAL) macros.
<b>&amp;SYS1.SFZHASRC</b>	BAL programs source.
<b>&amp;SYS1.SFZHDOCS</b>	HTML templates and CSS files.
<b>&amp;SYS1.SFZHJCLS</b>	JCL library with test programs.
<b>&amp;SYS1.SFZHLOAD</b>	Load library.
<b>&amp;SYS1.SFZHPROC</b>	PROC Library.

At installation, perform the tasks as described in these sections:

- “RMU default options (FZHOPTAB) table.”
- “RMU default library locator table (FZHPROCS)” on page 13.
- “Setting up the z/OS server (UNIX) environment” on page 14.
- “System information” on page 15.

On completion, tailor and test these jobs located in the &SYS1.SFZHJCLS library:

<b>FZHLINKJ</b>	Compile and link the RMU Script program.
<b>FZHRMUJ0</b>	Run RMU without the Script program.
<b>FZHRMUJ1</b>	Run the compiled Script program.
<b>FZHRMUJ2</b>	Run RMU Script as a “link and go” job.
<b>FZHDELJ</b>	Copy the test report with print control characters expanded.

If you are distributing HTML reports to the z/OS server (UNIX server on z/OS), tailor and test these jobs:

<b>FZHRMUX0</b>	Run RMU without the Script program.
<b>FZHRMUX1</b>	Run the compiled Script program.
<b>FZHRMUX2</b>	Run RMU Script as a “link and go” job.

Installation is successful if all jobs run to a normal end of job.

For instructions on how to use RMU for general users, see Chapter 2, “Working with RMU,” on page 3.

### RMU default options (FZHOPTAB) table

The RMU default options are located in the FZHOPTAB table. The source code for the table is located in the &SYS1.SFZHASRC library.

Users in the United States can use the table as shipped on the product tape.

Users outside the United States should adjust the currency and the decimal character by means of the CURRENCY= and the DECIMAL= parameters respectively.

## Installation

To assemble and link the table, use the FZHASMOJ job located in the &SYS1.SFZHJCLS library. You may need to tailor the job name and system libraries.

Figure 2 shows a copy of FZHOPTAB distributed with RMU:

---

```
FZHOPTAB FSMACRMU TYPE=E,          .E = GENERATE NUCLEUS          X
          OPSYS=MVS,              .OPERATING SYSTEM MVS        X
          COMPANY='FOUNDATION SOFTWARE INC.',          X
          DISTNUM=RMU02139,        .DISTRIBUTION ID FOR RMULIST  X
          RELEASE=V1R1.00,        .SOFTWARE RELEASE VERS,REL,MOD X
          RELDATE=02/01/09,        .SOFTWARE RELEASE DATE      X
          RMULIST=YES,            .TRANSLATOR DEFAULT LIST OPTION X
          COBLIST=YES,            .COBOL COMPILER DEFAULT LIST  X
          LKEDLST=YES,           .LINK STEP LIST OPTION       X
          DECIMAL=PERIOD,         .PERIOD/COMMA                X
          CURRENCY=$,             .CURRENCY SYMBOL              X
          CSVCHAR=',',            .DEFAULT CSV CHARACTER        X
          ERRLIMIT=128,           .ERROR LIMIT COUNTER          X
          TURF=1403-PAPER,        .TURF NO/1403-PAPER          X
          LINESIZE=512,           .MAXIMUM REPORT LINE SIZE (INCL CC) X
          PAGESIZE=66,           .MAXIMUM LINES PER PAGE (1 - 99) X
          MAXSIZE=32767,         .MAX FIELD LENGTH ALLOWED VIA DEFINE X
          FORMAT=HTML,           .DEFAULT FORMAT HTML/HTML1/CSV
          END FZHOPTAB
```

---

Figure 2. Copy of FZHOPTAB distributed with RMU

Explanation:

**TYPE=E** Macro type. Do not change.

**OPSYS=MVS** Operating system. Do not change.

**COMPANY='FOUNDATION SOFTWARE INC.'**  
Your company name up to 30 characters long.

**RELEASE=V1R1.00**  
Software release. Do not change.

**RELDATE=02/01/09**  
Software release date. Do not change

**RMULIST=YES**  
Default RMU Script compiler LIST option:  
**YES** Print RMU Script listing.  
**NO** Do not print RMU Script listing.

**COBLIST=YES**  
Default COBOL compiler LIST option:  
**YES** Print COBOL listing.

**LKEDLST=YES**  
Default Link LIST option:  
**YES** Print Link listing.  
**NO** Do not print Link listing.

**DECIMAL=PERIOD**  
Default character for decimal point:  
**PERIOD** Use period (.).  
**COMMA** Use comma (,).

<b>CURRENCY=\$</b>	Default currency symbol. The currency symbol must be a single character acceptable by COBOL.
<b>CSVCHAR=','</b>	Default separator character for CSV files. You can change this to a character of your own choosing.
<b>ERRLIMIT=128</b>	Error limit for the RMU Script compiler. The RMU Script compiler terminates after reaching the number of errors specified by this parameter.
<b>TURF=1403-PAPER</b>	Default background 1403-PAPER decorating option: <b>1403-PAPER</b> Generate HTML with stripes to simulate 1403 printer paper. <b>NO</b> Do not generate simulated 1403 printer paper.
<b>LINESIZE=512</b>	Maximum input reports line size (including print control character). This parameter is reserved for future use. Do not change.
<b>PAGESIZE=66</b>	Maximum input report lines per page. Do not change.
<b>MAXSIZE=32767</b>	Maximum RMU field size allowed. This size applies to the DEFINE statement in RMU Script. The maximum is 32,767. Do not change.
<b>FORMAT=HTML</b>	Default output format: <b>HTML</b> Produce a simple HTML report. <b>HTML1</b> Produce a HTML report with a selection tree on the left side by control break fields. RMU Script is required unless no control breaks are needed. <b>CSV</b> Produce a CSV file. Refer to the <b>CSVCHAR=</b> parameter for the default character to be used.

## RMU default library locator table (FZHPROCS)

RMU is compiled with the FZHRMU01 program.

FZHRMU01:

- Is invoked directly from JCL when the RMU Script program is to be compiled and linked only.
- Is invoked indirectly from the FZHRMU00 program when performing "link and go".
- Invokes and performs these tasks as a single step:
  - Converts RMU Script source to z/OS COBOL by means of the FZHRMUS1 utility.
  - Compiles the generated COBOL by means of z/OS Enterprise COBOL.
  - Links the compiled program.

To activate the FZHRMU01 program, perform these steps:

1. Tailor FZH#PROC, located in &SYS1.SFZHJCLS. This proc is read by the FZHRMU01 program. It contains all the information and steps needed to simulate the single step process.

## Installation

To tailor, follow the comments embedded in the proc. You will probably need to change these parameters: `TWORK=`, `CWORK=`, `MODEL=`, and `COBLIB=`. You may have to change other parameters, depending on your requirements. Verify the entire proc for potential changes.

If you comment out a keyword, you must also comment out all references to it. The `SYSPRDD=&DDname` option can be changed to redirect COBOL listings to a different DDname.

The default COBOL internal printer is set to `FJSYSPR`. This DD name is internally allocated and used by Migration Utility. Do not code it in the JCL.

While testing, you can change `SVC99=MSGOFF` in `FZH#PROC` to one of these options:

<b>MSGOFF</b>	Disable dynamic allocator tracing.
<b>MSGON</b>	Display JES messages.
<b>MSGALL</b>	Display input text and JES messages.
<b>MSGSER</b>	Display messages of a serious nature only.
<b>MSGTXT</b>	Display input text only.

Use these options to trace `FZH#PROC` and the dynamic allocation problems that might arise during testing. For production use, `SVC99=MSGOFF` is recommended to avoid excessive console messages.

**Note:** Messages can be controlled by placing this statement in the JCL:

```
//FJSVC99 DD DSN=&&FJSVC99(&SVC99),DISP=NEW
```

Where: `&SVC99` is one of the previously listed options.

2. Change the `FZHPROCS` table, located in `&SYS1.SFZHASRC`.

You must change this program to point to the RMU libraries installed on your system. Change the `PROCLIB0` constant to point to the PDS where `FZH#PROC` from step #1 is located. Change the `PRODUCT0` constant to the high-level qualifier of the Migration Utility libraries.

Assemble and link the `FZHPROCS` program using the `FZHASMPJ` job located in the `&SYS1.SFZHJCLS` library. This program must be linked into the RMU `&SYS1.SFZHLOAD` library.

**Note:** `FZHRMU01` loads `FZHPROCS` to locate the `FZH#PROC` member and to acquire a replacement for the `&SYS1` symbol located in `FZH#PROC`. The information in `FZHPROCS` must always point to proper libraries. This means that if you rename or move your RMU libraries, you must also adjust the information in `FZHPROCS` accordingly.

## Setting up the z/OS server (UNIX) environment

RMU can automatically publish the output documents to the z/OS server (UNIX) or to a regular z/OS variable-length flat file.

To publish to the z/OS server, the z/OS UNIX environment must be established and enabled. A root directory on the z/OS UNIX system must be established for each user. Coordinate access to UNIX environment with your z/OS system administrator.

RMU determines where to send the output by checking for the existence of the `FJUNIX0` DD name in the JCL. The `PATH=` parameter on the `FJUNIX0` definition points to the root directory showing where to write the documents.

If FJUNIX0 exists, RMU assumes that the HTML documents are being sent to the z/OS UNIX System. If FJUNIX0 does not exist, RMU creates a flat file on z/OS.

**Note:** UNIX files are handled by the FZHUNIX1 RMU program. This program is dynamically loaded at end of job when combining the output. UNIX is case-sensitive. That is, commands, directory, and file names must be typed exactly as defined on the UNIX system.

When publishing to the z/OS server, RMU must comply with the code set (ASCII or EBCDIC) defined for each document type on the UNIX system. The code set is obtained from the UNIX file, httpd.conf.

httpd.conf is pointed to by the FJCONFIG DD name in the JCL. It can be a z/OS PDS member, or a file in the UNIX directory.

If FJCONFIG is not in the JCL, ASCII is assumed for HTML and CSV files, and binary code for gif files.

These job examples located in &SYS1.SFZHJCLS can be used as templates to build custom jobs for publishing to z/OS UNIX:

FZHRMUX0  
FZHRMUX1  
FZHRMUX2

## System information

These jobs and JCL are distributed in the &SYS1.SFZHJCLS library:

<b>FZH#READ</b>	RMU system information.
<b>FZH#PROC</b>	RMU Script compiler proc.
<b>FZHASMOJ</b>	JCL to assemble the RMU options table.
<b>FZHASMPJ</b>	JCL to assemble the RMU default proc location.
<b>FZHDEL CJ</b>	Utility to expand the report print control characters.
<b>FZHLINKJ</b>	Job to compile and link RMU Script programs.
<b>FZHRMUJ0</b>	Job to run the FZHRMU00 utility (without Script program).
<b>FZHRMUJ1</b>	Job to run the FZHRMU00 utility with compiled script.
<b>FZHRMUJ2</b>	Job to run the FZHRMU00 utility with Script compile and "link and go".
<b>FZHRMUX0</b>	Job to run the FZHRMU00 utility (without Script program). Distributes to z/OS UNIX.
<b>FZHRMUX1</b>	Job to run the FZHRMU00 utility with compiled script. Distributes to z/OS UNIX.
<b>FZHRMUX2</b>	Job to run the FZHRMU00 utility with script compile, "link and go". Distributes to z/OS UNIX.
<b>FZHTEST0</b>	RMU Script example. Compile and link only.
<b>FZHTEST1</b>	RMU Script example. Compile, "link and go".
<b>FZHTEST2</b>	RMU Script example. Compile, "link and go".
<b>FZHTEST3</b>	RMU Script example. Compile, "link and go".
<b>FZHTEST4</b>	RMU Script example. Compile, "link and go".
<b>FZHRPT00</b>	Demo report file.

These procs are distributed in the &SYS1.SFZHPROC library:

Proc to compile and Link RMU Script.  
Proc to compile, "link and go", z/OS UNIX output.  
Proc to compile, "link and go", z/OS file output.  
Proc to run compiled script, z/OS UNIX output.  
Proc to run compiled script, z/OS file output.

## Installation

These members are distributed in the &SYS1.SFZHDOCS library:

<b>FZHBPARS</b>	Java program for parsing HTML docs on PC platform.
<b>FZHRMUL0</b>	RMU logo gif file.
<b>FZHMINUS</b>	Minus sign symbol.
<b>FZHPAGES</b>	Page break symbol.
<b>FZHPLUS0</b>	Plus sign symbol.
<b>FZHRARRW</b>	Arrow symbol.
<b>FZHSTOPO</b>	Stop symbol.
<b>FZHEBASE</b>	HTML template.
<b>FZHEBCSS</b>	Base CSS templates.
<b>FZHEINDEX</b>	JavaScript tree index.
<b>FZHELOAD</b>	JavaScript tree load.
<b>FZHEPCSS</b>	Tree CSS templates.

This BAL source is distributed in &SYS1.SFZHASRC:

<b>FZHOPTAB</b>	RMU options table.
<b>FZHPROCS</b>	RMU default proc locator.

This assembler macro is distributed in &SYS1.SFZHAMAC:

<b>FSMACRMU</b>	RMU Assembler macros needed for FZHOPTAB.
-----------------	---

These modules are distributed in the &SYS1.SFZHLOAD library:

<b>FZHATTCH</b>	RMU internal use. Task attach module.
<b>FZHCPYRT</b>	RMU internal use. Copyright information.
<b>FZHCSV00</b>	CSV file I/O module.
<b>FZHCVDAT</b>	RMU internal use. Date retrieval module.
<b>FZHDDCPY</b>	RMU internal use. Dynamic allocator submodules.
<b>FZHDELCC</b>	Utility program to expand report print control characters.
<b>FZHDYNCV</b>	RMU internal use. EBCDIC/ASCII code conversion module.
<b>FZHJOB0</b>	RMU internal use. Get job information.
<b>FZHGOPT0</b>	RMU internal use. Dynamic allocator submodules.
<b>FZHGPRM0</b>	RMU internal use. PARM retrieval from EXEC statement.
<b>FZHGUSER</b>	RMU internal use. Registration module.
<b>FZHHTML0</b>	RMU internal use. HTML formatter module.
<b>FZHHTML1</b>	RMU internal use. HTML1 formatter module.
<b>FZHJCL00</b>	RMU internal use. Installation JCL tailoring utility.
<b>FZHJCL01</b>	RMU internal use. Installation JCL tailoring utility.
<b>FZHMVSC0</b>	RMU internal use. Console I/O module.
<b>FZHMVSC1</b>	RMU internal use. Console I/O module.
<b>FZHOPTAB</b>	RMU options table. Prepared by the installer.
<b>FZHPROCS</b>	RMU default proc locator. Prepared by the installer.
<b>FZHRMUB1</b>	RMU internal use. COBOL skeleton.
<b>FZHRMUSP</b>	RMU internal use. I/O module for RMU listing.
<b>FZHRMUS1</b>	RMU internal use. RMU script compiler submodule.
<b>FZHRMUS2</b>	RMU internal use. RMU script compiler submodule.
<b>FZHRMUS3</b>	RMU internal use. RMU script compiler module.
<b>FZHRMUW1</b>	RMU internal use. COBOL and RMU reserved words.
<b>FZHRMU00</b>	RMU main utility for "link and go" or run without RMU script.
<b>FZHRMU01</b>	RMU Script compiler one step driver.
<b>FZHSRC00</b>	RMU developers tool.
<b>FZHSTAE0</b>	RMU compiler abend interrupt handler.
<b>FZHSVC99</b>	Dynamic Allocator program (SVC99 handler).
<b>FZHTEST0</b>	RMU compiled script test program.
<b>FZHUNIX0</b>	RMU internal use. Z/OS UNIX interface module.
<b>FZHUNIX1</b>	RMU internal use. Z/OS UNIX interface module.



---

## Chapter 4. Working with RMU Script

This section describes in detail how to work with the RMU Script language

---

### Basic concepts

RMU Script consists of RMU Script language statements (collectively a script program) that manipulate the input and formats the output document.

The script program is prepared with the ISPF editor in a PDS, as a regular text file. The program must be compiled with the RMU Script compiler and linked as a load module, or linked and executed (run as "link and go"), depending on the JCL used.

The RMU Script compiler translates script statements to COBOL; therefore a z/OS COBOL or LE COBOL compiler is required.

An RMU Script program consists of 10 optional sections as shown here. You must supply at least one section to make it a valid program.

<b>Environment section</b>	The PARM statement with its options. If coded, PARM must be the first statement in the program.
<b>Working Storage section</b>	Defines the fields and variables used in the Activity section.
<b>&lt;style&gt; section</b>	Defines the Cascaded Style Sheets (color, fonts, and so on).
<b>&lt;object&gt; section</b>	Defines the images to be inserted in the output document.
<b>&lt;docs_top&gt; section</b>	Defines the images and text to be inserted at the top of output document.
<b>&lt;docs_end&gt; section</b>	Defines the images and text to be inserted at the bottom of output document.
<b>&lt;page_top&gt; section</b>	Defines the images and text to be inserted at the top of each page in the output document.
<b>&lt;page_end&gt; section</b>	Defines the images and text to be inserted at the bottom of each page in the output document.
<b>Activity section</b>	Contains the language statements that filter the input and construct the output documents.
<b>Control section</b>	Defines control breaks for HTML1 format documents.

---

### I/O handling

Files cannot be defined in RMU Script. I/O is completely concealed from you. One input report and one output document are assumed.

RMU Script activity statements operate on the page buffer. RMU reads in one report page at the time. In this way, all lines on a single page can be accessed as a single resource. Each page contains up to 66 lines. The activity statements are executed from top to bottom as coded in the program.

## I/O handling

The report page begins with Channel 1. That is, a "1" in the first position is Channel 1. If there are more than 66 lines on each page, the remaining lines are drained to the output document unchanged. The activity logic is not applied to the remaining lines.

Information on each line is accessed by explicitly coding the line number with substring notation. For example, LINE1 is line 1, LINE2 is line 2 ... LINE66 is line 66. This example evaluates line 1 starting in position 3 for 5 bytes:

```
IF LINE1 (3: 5) = 'abcde'
```

If the line number is not specified, then a subscript named IDX is used to subscript the lines. IDX is a reserved variable, but it can be assigned to a desired line number to be accessed.

**Note:** The input report must be a valid printer file with the print control character in position 1. RMU expands print control characters to compensate for the blank lines that would normally be seen on a printed report. Therefore, when referring to a specific line by number, the line number is as according to the expanded report.

These are recognized control characters:

1	Channel 1.
(blank)	Space 1.
0	Space 2.
-	Space 3.
+	Suppress space (for RMU, this is the same as space 1).

When coding a Script program, run the FZHDELJC utility to expand the report control characters. This will help to identify the line numbers of interest. The FZHDELJC JCL is located in the &SYS1.SFZHJCLS library.

---

## Coding rules

These coding rules apply:

- RMU Script statements can be placed anywhere between columns 1 and 72. Only one statement is allowed on a line. Statements are followed by respective arguments.
- The arguments are separated by one or more spaces.
- Lines that begin with an asterisk (\*) are treated as comments.
- Comments can be imbedded anywhere in the program.
- Alphanumeric constant values (literal) are enclosed in quotes. For example:  
'abcd'
- A hex literal is coded by placing an X before the hex constant. For example:  
X'1234'
- A hex literal must contain a combination of valid hex characters: 0–9, A–F.
- Numeric constant values are coded as numbers with a leading plus or minus sign and, if necessary, a decimal point. Numbers without a sign are treated as positive numbers. For example, -123.55 is a negative number.

## Program example

The program shown in Figure 3 reads in the FZHRPT00 test report located in the &SYS1.SFZHJCLS library and produces an HTML1-format document with a control breaks tree on the left side and the report body on the right side of the screen.

The background of the output document simulates 1403 printer paper.

A page break image, fzhpages.gif, is inserted at the top of each page.

When LINE8-LINE55 contains 'DDDD' in position 24 for 4, the value is made red (.em1). When LINE8-LINE55 contains 'EBEE' in position 24 for 4, the value is made blue (.em2).

PARM LIST COBOL LKED FORMAT HTML1 TURF 1403-paper	Environment Section
*****	
* This program converts a report to RMU HTML1 format.	* Comments
* Some decorating is inserted via styles and objects.	*
*****	
DEFINE COMPANY W 2 A	Working Storage Section
DEFINE BRANCH W 5 A	
DEFINE OFFICER W 4 A	
<style>	
.em1 {color: red;}	Style Section
.em2 {color: blue;}	
</style>	
<page_top>	page_top Section
<IMAGE SRC="images/fzhpages.gif">	
</page_top>	
IF (LINE5 (2: 7) = 'COMPANY')	
AND (LINE5 (13: 6) = 'BRANCH')	
AND (LINE8 (4: 2) IS NUMERIC)	Activity
COMPANY = LINE8 (4: 2)	
BRANCH = LINE8 (14: 3)	
OFFICER = LINE8 (24: 4)	
EVALUATE LINE8-LINE55 (24: 4)	Section
WHEN 'DDDD'	
LINE (24: 4) = .em1	
WHEN 'EBEE'	
LINE (24: 4) = .em2	
END-EVALUATE	
END-IF	
CONTROL COMPANY BRANCH OFFICER	Control Section

Figure 3. Example of RMU script program to produce an HTML1 format document

Figure 4 on page 20 shows a screen print of the generated HTML1 document:



---

## Creating CSV documents

When creating a CSV document, extraneous information such as page titles, field headings, and control break annotations must be removed from the input.

Additionally, you may want to remove float characters from the numeric fields and re-arrange the sign such that the spreadsheet program can intelligently format numeric cells.

Finally, you need to insert a special character between each field to delineate columns for import.

RMU provides methods and statements specifically designed to make these tasks simple.

- To create a CSV file, code `FORMAT CSV='&chr'` on the PARM statement where: `&chr` is the separator character to be placed between the fields.
- Use these RMU methods in the assign statement to format column values:
  - `.ETEXT` For alphanumeric fields.
  - `.ENUM1` For numeric and numeric edited fields.
- Use `PAGE-COUNT` and `LINE-COUNT` to determine the page properties.
- Use the `BYPASS` statement to remove a specific line or a range of lines.
- Use the `IDX` subscript to control a `DO` loop for a specific number of lines.

The line is selected for output if at least one method is applied, otherwise the line is bypassed.

**Note:** CSS (color, fonts, and so on) and images cannot be applied to a CSV file.

The RMU Script program shown in Figure 5 on page 22 creates a CSV file from the FZHRPT00 report located in `&SYS1.SFZHJCLS`. The source code is FZHTEST4 located in the `&SYS1.SFZHJCLS` library.

## Creating CSV documents

---

```
PARM LIST COBOL LKED FORMAT CSV=':' DECIMAL (PERIOD) CURRENCY($)
*****
* This program converts test report to RMU CSV file format.          *
*                                                                    *
* The program demonstrates how to trim un-needed report lines and  *
* strings from a report. The output is a CSV file ready for import  *
* into a Spreadsheet.                                              *
*                                                                    *
* The trimming is done via the .ETEXT and .ENUM1 RMU methods      *
*****
IF (PAGE-COUNT > 1)
* REPORT TITLES ARE BYPASSED ON ALL BUT FIRST PAGE.
  BYPASS LINE1-LINE7
ELSE
* PRESERVE THE FIRST PAGE TITLE LINES FOR spreadsheet
  LINE1 (2: 80) = .ETEXT
  IDX = 1
  DO 6 TIMES
    IDX = (IDX + 1)
    LINE (02: 07) = .ETEXT
    LINE (13: 06) = .ETEXT
    LINE (23: 07) = .ETEXT
    LINE (34: 14) = .ETEXT
    LINE (52: 06) = .ETEXT
    LINE (62: 10) = .ETEXT
  END-DO
END-IF
IDX = 7
  * Line 8 to LINE-COUNT are detail lines on every page
DO WHILE (IDX < LINE-COUNT)
  IDX = (IDX + 1)
* get rid of Control break and FINAL lines
  IF (LINE (2: 5) = 'FINAL')
  OR (LINE (2: 6) = 'FILEIN')
  OR (LINE (24: 4) = SPACES)
    BYPASS LINE
  ELSE
* construct a row of character separated values
    LINE (02: 07) = .ETEXT
    LINE (13: 06) = .ETEXT
    LINE (23: 07) = .ETEXT
    LINE (34: 14) = .ENUM1
    LINE (52: 06) = .ENUM1
    LINE (62: 10) = .ENUM1
  END-IF
END-DO
```

---

Figure 5. Example of RMU script program to create a CSV file

Figure 6 on page 23 shows the first few records created by the program in Figure 5:

---

```

4/08/08      COMPANY= 10  THIS IS A TEST REPORT-1                PAGE      1
:           :           :           :           :           :
:           :           :           :           :           :
:           :           :           :           :           :
COMPANY : BRANCH : OFFICER :           :           :           :
C       : NUMBER : NUMBER  : WAGE      : RATE     : BONUS    :
:       :       :         :           :           :           :
10      : 001   : DDDD   : 57500.00 : 10.900  : 6267    :
:       :       : EBEE   : 60000.00 : 11.000  : 6600    :
:       :       : EBEE   : 60000.00 : 11.000  : 6600    :
:       :       : EBEE   : 60000.00 : 11.000  : 6600    :
:       :       : EBEE   : 60000.00 : 11.000  : 6600    :
:       :       : EBEE   : 60000.00 : 11.000  : 6600    :

```

---

Figure 6. Records created by the example program shown

---

### Creating simple HTML documents

A simple-HTML format document can be created without the use of RMU Script. To do this, use the FZHRMUJ0 JCL located in the &SYS1.SFZHJCLS library.

You can create a simple HTML document with special decorating if needed, such as special colors, fonts and images, as outlined in this section:

- Code FORMAT HTML on the PARM statement.
- Optionally, code <style>, <object>, <docs\_top>, < docs\_end>, <page\_top> and <page\_end> sections to define decorating. Use <style> and <object> methods to decorate text. You do so by assigning method names to line text in the Activity section.

**Note:** Images are not automatically included in the output document; therefore, if images are used, they must be available on the server where you place the document.

- Use PAGE-COUNT and LINE-COUNT to determine the page properties.
- Use the BYPASS statement to remove a specific line or a range of lines.
- Use the IDX subscript to control a DO loop for a specific number of lines.
- Use the EVALUATE statement to loop through a range of lines when a specific location on multiple lines is to be decorated with different methods.

The RMU Script program shown in Figure 7 creates a simple HTML document from the FZHRPT00 report located in the &SYS1.SFZHJCLS library. The source code is in FZHTEST1 located in the &SYS1.SFZHJCLS library.

---

```
PARM LIST COBOL LKED FORMAT HTML turf 1403-paper
*****
* This program converts test report to RMU HTML format.          *
* Some decorating is inserted via style and object methods.      *
*****
<style>
  .em1 {color: red; font-weight: normal;}
  .em2 {color: blue;}
  .em3 {color: orange;}
</style>
<page_top>
<IMAGE SRC="images/fzhpages.gif">
</page_top>
IDX = 1
IF (LINE5 (2: 7) = 'COMPANY')
AND (LINE5 (13: 6) = 'BRANCH')
AND (LINE8 (4: 2) IS NUMERIC)
  EVALUATE LINE8-LINE55 (24: 4)
    WHEN 'ZZZZ'
      LINE (1: 80) = .EM1
    WHEN 'EBEE'
      LINE (24: 4) = .EM2
    WHEN OTHER
      LINE (24: 4) = .EM3
  END-EVALUATE
END-IF
```

---

Figure 7. Example of RMU script program to create a simple HTML document

Figure 8 on page 25 shows a screen print of the generated HTML document:





### Creating HTML1 documents with the control breaks selection tree

To create an HTML1-format document with special decorating such as special colors, fonts and images, create an RMU Script as outlined in this section:

- Code `FORMAT HTML1` on the `PARM` statement.
- Use the `DEFINE` statement to define working storage fields for control breaks and other needs.
- Optionally, code `<style>`, `<object>`, `<docs_top>`, `<docs_end>`, `<page_top>` and `<page_end>` sections to define decorating. Use `<style>` and `<object>` methods to decorate text. To do this, you assign method names to line text in the Activity section.

**Note:** Images are automatically included in the output document if located in the `FJIDOC0` library.

- Optionally use `PAGE-COUNT` and `LINE-COUNT` to determine the page properties.
- Optionally use the `BYPASS` statement to remove a specific line or a range of lines.
- Optionally use the `IDX` subscript to control a `DO` loop for a specific number of lines.
- Optionally use the `EVALUATE` statement to loop through a range of lines when a specific location on multiple lines is to be decorated with different methods.
- Use the `CONTROL` statement to declare control break fields. Note that this is a required statement.

Control break fields must be carefully populated from the supplied information in each page. You will probably have to test lines for certain values, such as field titles to make sure that the correct values are used. See Figure 9 on page 27.

The program shown in Figure 9 on page 27 reads in the `FZHRPT00` test report located in the `&SYS1.SFZHJCLS` library and produces an HTML1 format document with a control breaks tree on the left side and the report body on the right side of the screen. The source code is in `FZHTEST2` located in the `&SYS1.SFZHJCLS` library.

The background of the output document simulates 1403 printer paper. A page break image, `fzhpages.gif`, is inserted at the top of each page.

When `LINE8-LINE55` contains `'DDDD'` in position 24 for 4, the value is made red (`.em1`). When `LINE8-LINE55` contains `'EBEE'` in position 24 for 4, the value is made blue (`.em2`).

## Creating HTML1 documents with the control breaks selection tree

---

```
PARM LIST COBOL LKED FORMAT HTML1 TURF 1403-paper
*****
* This program converts a report to RMU HTML1 format.          *
* Some decorating is inserted via styles and objects.          *
*****
DEFINE COMPANY W 2 A
DEFINE BRANCH W 5 A
DEFINE OFFICER W 4 A
<style>
  .em1 {color: red;}
  .em2 {color: blue;}
</style>
<page_top>
<IMAGE SRC="images/fzhpages.gif">
</page_top>
IF (LINE5 (2: 7) = 'COMPANY')
AND (LINE5 (13: 6) = 'BRANCH')
AND (LINE8 (4: 2) IS NUMERIC)
  COMPANY = LINE8 (4: 2)
  BRANCH = LINE8 (14: 3)
  OFFICER = LINE8 (24: 4)
  EVALUATE LINE8-LINE55 (24: 4)
    WHEN 'DDDD'
      LINE (24: 4) = .em1
    WHEN 'EBEE'
      LINE (24: 4) = .em2
  END-EVALUATE
END-IF
CONTROL COMPANY BRANCH OFFICER
```

---

Figure 9. Example of RMU script program to produce an HTML1 format document

Figure 10 on page 28 shows a screen print of the generated HTML1 document:



## Decorating an HTML document

The output document can be decorated in three ways:

- **Images and special text can be created at the top of the document, the bottom of the document, the top of each page, and the bottom of each page.**

File types recognized as images by RMU files are: JPG, TIF, PSD, PDD, BMP, PNG, and GIF.

To do this, you code valid HTML between these tags:

**<docs\_top> </docs\_top>**

This optional section defines the images and text to be inserted at the top of the document.

**<docs\_end> </docs\_end>**

This optional section defines the images and text to be inserted at the bottom of the document.

**<page\_top> </page\_top>**

This optional section defines the images and text to be inserted at the top of each page.

**<page\_end> </page\_end>**

This optional section defines the images and text to be inserted at the bottom of each page.

RMU uses `<pre>` and `</pre>` HTML tags to preserve the original document format, including spacing and blank lines. These tags tell HTML to leave the text pre-set as it is. Therefore when coding text, you can type the text without the special tags, or you can put the text between the `</pre>` and `<pre>` tags to resume standard default HTML formatting.

This example shows both lines at the top of the document:

```
<docs_top>
  RMU generated document
  as of 11/15/2008
</docs_top>
```

This example shows text as a single line at the top of the document:

```
<docs_top>
  </pre>
  RMU generated document
  as of 11/15/2008
  <pre>
</docs_top>
```

Images included in the HTML text placed between the tags are automatically resolved by RMU if they exist in the FJIDOC0 library.

For example, the code shown here causes the xyzimag1 and gif file to be included in the document from the FJIDOC0 file because of `SRC="images/xyzimag1.gif"`:

```
<docs_top>
  <IMAGE SRC="images/xyzimag1.gif">
  RMU generated document as of 11/15/2008
</docs_top>
```

RMU does not validate any information included between the tags. Errors are discovered when you browse the document.

These images are available in the `&SYS1.SFZHDOCS` library:

<b>FZHMINUS</b>	Minus sign.
<b>FZHPAGES</b>	Page break line.
<b>FZHPLUS0</b>	Plus sign.
<b>FZHRARRW</b>	Arrow.

## Decorating an HTML document

FZHRMUL0 RMU logo.

FZHSTOPO Stop sign.

- **Special fonts and colors can be applied to line text conditionally.**

The best way of doing this is to declare styles (CSS) in the <style> section and then assign the styles to the text in question on each line.

You can choose to highlight special text. For example, title lines can be made a different color, negative amounts can be turned red, and so on.

Changing the font size is not recommended as it may cause text alignment problems.

**Note:** When declaring a style in the <style> section, the style parameters are not validated. Errors are discovered when you browse the document.

In this example, the DDDD value on line 8 through line 55 starting in position 24 is made red and EBEE is made blue:

```
<style>
  .em1 {color: red;}
  .em2 {color: blue;}
</style>
EVALUATE LINE8-LINE55 (24: 4)
  WHEN 'DDDD'
    LINE (24: 4) = .em1
  WHEN 'EBEE'
    LINE (24: 4) = .em2
END-EVALUATE
```

Likewise, the IF statement can be used to do the same.

- **Simulated 1403 printer paper background turf is the default.**

The turf can be turned off by specifying TURF NO on the PARM statement at the top of the script program. The default background color will be as defined for the body in the FZHEBCSS member located in the &SYS1.SFZHDOCS library. You can override the default by adding your own body CSS in the <style> section.

**Example:**

```
PARM TURF NO
<style>
  body {color: black; Background-color: white; font-size: &size%;}
</style>
```

---

## Debugging RMU Script programs

RMU generates COBOL programs that are compiled with standard IBM® COBOL, therefore any runtime problems that you experience are as for any other non-RMU COBOL programs.

Compile RMU Script with the PROCESS LIST,MAP COBOL options. A PROCESS statement can be placed before the PARM statement. Note that multiple PROCESS statements can be coded.

The RMU-generated COBOL contains statement sequence numbers in columns 1 to 6.

There are 3 types of sequence numbers:

**#nnnnn** Statement is generated according to the *nnnnn* RMU script statement number.

**#BASE** Statement is generated from the COBOL skeleton.

**#EOF** Statement is generated after the last RMU Script statement is processed.

COBOL runtime problems fall into one of these categories:

- **File I/O errors**

File problems such as wrong LRECL, or no DD in the JCL. RMU intercepts I/O errors and issues an appropriate message. Look at the console and SYSOUT messages for more information.

- **Environmental errors and ABENDs**

Errors associated with such things as JCL issues, RACF®/security problems, and memory problems. Such problems are intercepted and logged by the operating system. In most instances, the job is canceled by the operating system with an appropriate message. Look at the console and SYSOUT messages for more information.

- **Program check interrupts**

Processing problems such as an addressing exception, data exception, or specification exception.

To locate the RMU Script program statement in error:

1. If you have a COBOL debugger, use the debugger report to find the COBOL statement in error.
2. If you do not have a debugger:
  - In the Log or Dump file, find the hex offset of the PSW in your program as displayed by the Operating System.
  - Perform a find on the offset address in your COBOL compiler listing (expanded Assembler listing).
  - The COBOL source statement or line number can be found in the area before the offset.
  - Look at the COBOL listing identified by the line number in error.
3. The RMU Script statement number is in columns 1-6.
  - If #nnnnn is shown, this is the RMU Script statement in error.
  - If '#EOF' or '#BASE' is shown, look backward at statement numbers until you find a #nnnnn or a paragraph name (whichever you find first).
  - For a paragraph name, trace back to the perform statement to locate the routine that invoked it, then repeat the previous step.
4. Determine the cause of the program check interrupt from the located statement. If a protection exception occurred, check for bad subscripts. If a data exception occurred, check for non-numeric data in arithmetic operations.

## Debugging RMU Script programs



---

## Chapter 5. RMU Script language instruction reference

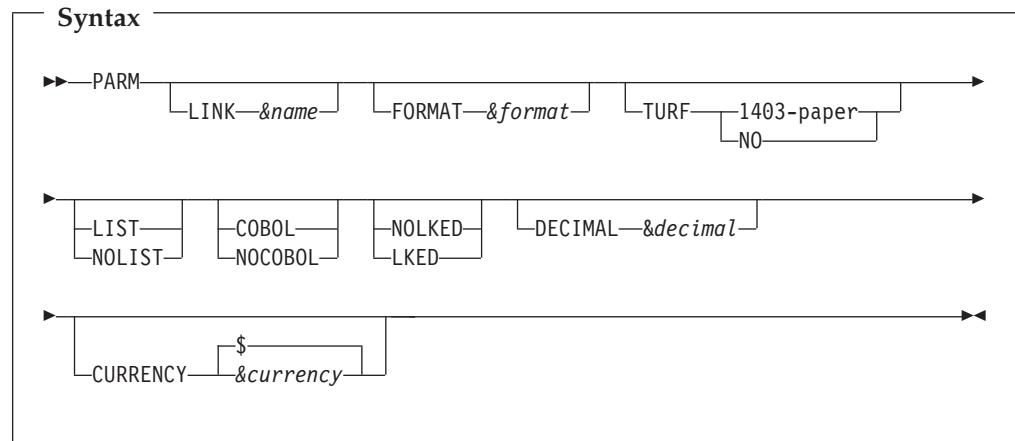
This section lists program instructions, syntax, further explanation, and examples, for each instruction supported by RMU.

---

### PARAM statement

The PARAM statement defines the RMU Script language compiler options. These options override the defaults in the FZHOPTAB options table.

#### Syntax



**Note:** Options are coded on the same line following the PARAM keyword. If all options do not fit on a single line, multiple PARAM lines can be coded as needed.

#### Parameters

<i>&amp;name</i>	Program name 1 to 8 characters long. The name must be a valid COBOL program name. The default is NONAME.
<i>&amp;format</i>	Type of output. Valid values are: <b>HTML0</b> Output is simple HTML format. <b>HTML1</b> Output is HTML format with a directory of control break fields on the left. <b>CSV='&amp;char'</b> Output is a CSV file, where <i>&amp;char</i> is the value separator character for CSV files.
<b>1403-paper</b>	Simulate green-striped 1403 printer paper background.
<b>NO</b>	Use gray background color.
<i>&amp;decimal</i>	Decimal point character. Valid values are: <b>PERIOD</b> Use '.' for decimal point. <b>COMMA</b> Use ',' for decimal point.
<i>&amp;currency</i>	Character to be used as currency symbol.

## PARM statement

### Examples

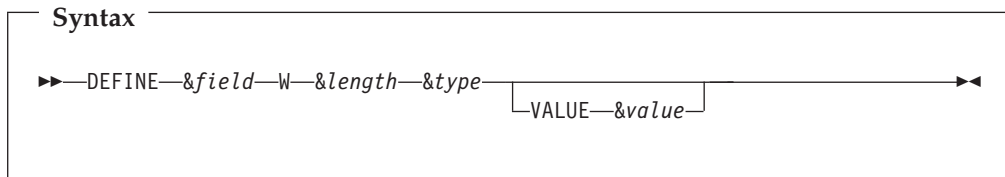
```
PARM LIST COBOL LKED FORMAT HTML1 TURF 1403-paper  
PARM LINK FZHTST0 LIST COBOL LKED FORMAT HTML1 TURF 1403-paper
```

---

## DEFINE statement

The DEFINE statement defines the RMU Script program variables and field names to be used within the program.

### Syntax



### Parameters

- &field* Variable name 1 to 16 characters long. The name must begin with an alpha character. The name can be composed of letters, numbers, and a minus sign ('-').
- Example of valid names:  
WS-BALANCE  
WS-AMOUNT
- &length* Field length in bytes. Maximum length for N fields is 18. Maximum length for P fields is 9. Maximum length for alpha fields is 32767.
- &type* Field type. Valid values are:  
N Unsigned numeric display integer.  
A Alphanumeric.  
B Signed binary integer, 2- or 4-byte binary integer.  
P Signed packed decimal number.
- &value* Initial value. An alpha value must be enclosed in quotes. a numeric value must not exceed the field length.
- These reserved values can be coded:  
SPACES or SPACE  
ZEROS or ZEROES or ZERO  
LOW-VALUES or LOW-VALUE  
HIGH-VALUES or HIGH-VALUE

### Programming notes

RMU does not support decimal places at this time.

### Examples of field definitions

```
DEFINE WX-WORK W 10 A  
DEFINE WX-NUM W 5 N  
DEFINE WX-PACK W 5 P  
DEFINE WX-BIN2 W 2 B  
DEFINE WX-BIN4 W 4 B
```

```

DEFINE WS-WORK W 10 A VALUE 'ABCDEF'
DEFINE WS-NUM W 5 N VALUE 12345
DEFINE WS-PACK W 5 P VALUE +123456789
DEFINE WS-BIN2 W 2 B VALUE 1234
DEFINE WS-BIN4 W 4 B VALUE 123456789

```

## <object> and </object> tag

The <object> tag declares the beginning of object and image definitions to be inserted in the output document. The object section begins with the <object> tag and terminates with the </object> tag. One or more images can be declared between the tags.

```

<object>
    .im&mod {<IMAGE SRC="images/&image.gif">}
    :
</object>

```

## Parameters

*&mod* Image name modifier. The name is composed of .im&mod. For example, when &mod is "g1", the name is .img1. This name is referenced in the assign statement within the RMU Script program.

*&image* The image file name as found in the FJIDOC0 library. For example, fzhpages and fzhplus0 are gif files located in the &SYS1.SFZHDOCS default FJIDOC0 library. To declare these two gif files for use in an RMU Script program, use the code:

```

<object>
    .img1 {<IMAGE SRC="images/fzhpages.gif">}
    .img2 {<IMAGE SRC="images/fzhplus0.gif">}
</object>

```

## Programming notes

When the output is directed to z/OS UNIX, images are copied from the &SYS1.SFZHDOCS file in binary format to the r001\images directory.

When the output is directed to the z/OS flat file for download to a PC or a server, images are included in the download document and parsed on the target PC or server with the FZHBPARS Java utility.

Images are assigned to a specific location in the document. Unresolved images are ignored.

### Example:

```

LINE1 (10: 0) = .img1

```

Inserts img1 before position 10 on line 1.

## <style> and </style> tag

The <style> tag declares the beginning of Cascaded Style Sheets (CSS) to be applied to the output text. The style section begins with the <style> tag and terminates with the </style> tag. One or more styles can be declared between the tags.

There are three distinct styles that can be declared:

## <style> and </style> tag

- Style for document body.
- Colors for .turf1 and .turf2 alternated to simulate 1403 green-striped printer paper.
- Styles for highlighting and decorating text (element styles).

```
<style>
  body {color: &color; Background-color: &color; font-size: &size%;}
  .turf1 {Background-color: &color}
  .turf2 {Background-color: &color}
  .em&n {color: &color;}
:
</style>
```

### Parameters

<i>&amp;color</i>	A valid HTML color.
<i>&amp;size</i>	Font size as an absolute size or percentage.
<i>&amp;n</i>	Element name modifier.

### Programming notes

All styles are optional.

Body and Element styles can include any combination of values acceptable to HTML. The content enclosed in { } is not validated by MU, therefore special care must be taken to make sure that the specified options are correct. Once defined, the body style is automatically applied to the document body.

.turf1 and .turf2 are special element styles for simulating 1403 printer paper. Color is the only acceptable option. These options are used automatically when the 1403-paper option is in effect. .turf1 and .turf2 colors are alternated to simulate 1403 printer paper.

.em&n styles are for decorating report text. Once defined, .em&n styles can be assigned to specific fields in the document. For example, negative amounts can be turned red and so on. One or more element styles can be defined.

### Examples

```
<style>
  body {color: green; Background-color: light-gray; font-size: 100%;}
  .turf1 {Background-color: ccfff1}
  .turf2 {Background-color: 99fff1}
  .em1 {color: red;}
  .em2 {color: blue;}
</style>
```

---

## <docs\_top> and </docs\_top> tag

The <docs\_top> tag declares the beginning of the text and the images to be inserted at the top of the output document. The docs\_top section begins with the <docs\_top> tag and terminates with the </docs\_top> tag.

You must code valid HTML strings between the tags. The content between the tags is placed at the top of the document without validation. Errors are not discovered until the document is browsed.

```
<docs_top>
:
</docs_top>
```

**Parameters**

None.

**Programming notes**

Images included in the HTML text placed between the tags are automatically resolved by RMU if they exist in FJIDOC0 library.

For example, the code shown here causes the xyzimag1 and xyzimag2 gif files to be included in the document from the FJIDOC0 file:

```
<docs_top>
  <IMAGE SRC="images/xyzimag1.gif">
  <IMAGE SRC="images/xyzimag2.gif">
</docs_top>
```

**<docs\_end> and </docs\_end> tag**

The <docs\_end> tag declares the beginning of the text and images to be inserted at the bottom of the output document. The docs\_end section begins with the <docs\_end> tag and terminates with the </docs\_end> tag.

You must code valid HTML strings between the tags. The content between the tags is placed at the bottom of the document without validation. Errors are not discovered until the document is browsed.

```
<docs_end>
:
</docs_end>
```

**Parameters**

None.

**Programming notes**

Images included in the HTML text placed between the tags are automatically resolved by RMU if they exist in FJIDOC0 library.

For example, the code shown here causes the xyzimag1 and xyzimag2 gif files to be included in the document from the FJIDOC0 file:

```
<docs_end>
  <IMAGE SRC="images/xyzimag1.gif">
  <IMAGE SRC="images/xyzimag2.gif">
</docs_end>
```

**<page\_top> and </page\_top> tag**

The <page\_top> tag declares the beginning of the text and images to be inserted at the top of each page in the output document. The page\_top section begins with the <page\_top> tag and terminates with the </page\_top> tag.

You must code valid HTML strings between the tags. The content between the tags is placed at the top of each page in the document without validation. Errors are not discovered until the document is browsed.

## <page\_top> and </page\_top> tag

```
<page_top>  
:  
</page_top>
```

### Parameters

None.

### Programming notes

Images included in the HTML text placed between the tags are automatically resolved by RMU if they exist in FJIDOC0 library.

For example, the code shown here causes the xyzimag1 and xyzimag2 gif files to be included in the document from the FJIDOC0 file:

```
<page_top>  
  <IMAGE SRC="images/xyzimag1.gif">  
  <IMAGE SRC="images/xyzimag2.gif">  
</page_top>
```

---

## <page\_end> and </page\_end> tag

The <page\_end> tag declares the beginning of the text and images to be inserted at the bottom of each page in the output document. The page\_end section begins with the <page\_end> tag and terminates with the </page\_end> tag.

You must code valid HTML strings between the tags. The content between the tags is placed at the bottom of each page in the document without validation. Errors are not discovered until the document is browsed.

```
<page_end>  
:  
</page_end>
```

### Parameters

None.

### Programming notes

Images included in the HTML text placed between the tags are automatically resolved by RMU if they exist in FJIDOC0 library.

For example, the code shown here causes the xyzimag1 and xyzimag2 gif files to be included in the document from the FJIDOC0 file:

```
<page_end>  
  <IMAGE SRC="images/xyzimag1.gif">  
  <IMAGE SRC="images/xyzimag2.gif">  
</page_end>
```

---

## Assignment statement

The assignment statement assigns a value to a field. The value can be another field, a literal, a style, a method, an image, or an arithmetic expression.

There are four types of assignment statements:

### Normal assignment

Assigns field values and arithmetic outcomes to a field.

**Style assignment to a line or a part of a line**

Assigns fonts and colors (CSS) for decorating line text. The style to be inserted must be defined in the <style> section.

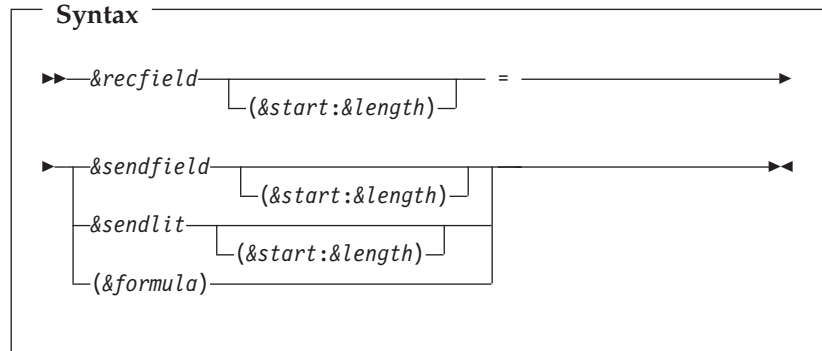
**Method assignment to line text (columns) when creating CSV files**

De-edits numeric values and prepares line text for spreadsheet use.

**Object assignment to a line or line text**

Inserts objects (images) in a specific position on the report line. The objects to be inserted must be defined in the <object> section.

**Format 1 Assignment**



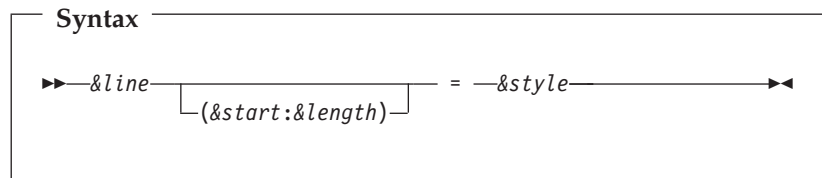
**Parameters:**

- &recfield* Specifies the field name to which the value will be assigned.
- &start* Starting position.
- &length* Length to be moved.
- equal sign (=) Indicates assignment.
- &sendfield* Sending field (field to be copied).
- &sendlit* Sending value can be a literal. An alphanumeric literal must be enclosed in quotes.
- &formula* Arithmetic expression. It can contain arithmetic operators (+, -, \*, /). The outcome of the calculation is placed in the *&recfield*.

**Examples:**

```
WS-COMPANY = LINE5 (5: 4)
WS-COMPANY = '1234'
WS-COMPANY = '1234567890 (3:4)
WS-INTEGGER = (WS-INTEGGER + 1)
```

**Format 2 Assignment**



**Parameters:**

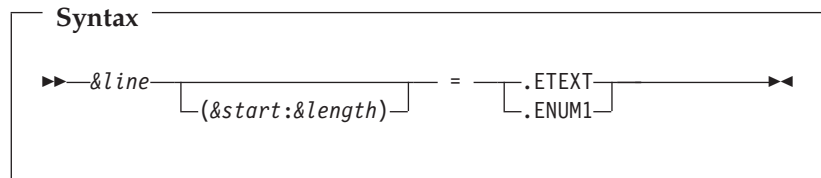
## Assignment statement

<i>&amp;line</i>	Specifies the line to which the value will be assigned.
<i>&amp;start</i>	Starting position.
<i>&amp;length</i>	Text length <i>&amp;style</i> applies to.
equal sign (=)	Indicates assignment.
<i>&amp;style</i>	A style name as declared in the <style> section.

### Example:

```
LINE10 (15:6) = .em1
```

### Format 3 Assignment



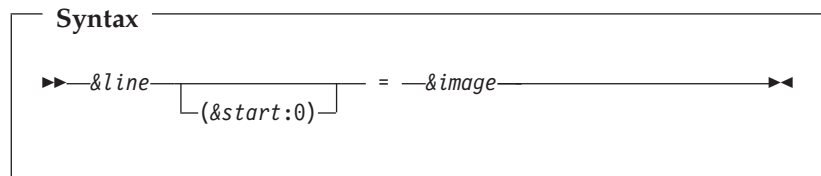
### Parameters:

<i>&amp;line</i>	Specifies the line to which the value will be assigned.
equal sign (=)	Indicates assignment.
<i>&amp;start</i>	Starting position.
<i>&amp;length</i>	Text length <i>&amp;style</i> applies to.
<b>.ETEXT</b>	Create plain text field.
<b>.ENUM1</b>	De-edit numeric field.

### Examples:

```
LINE10 (15:6) = .ETEXT  
LINE10 (15:6) = .ENUM1
```

### Format 4 Assignment



### Parameters:

<i>&amp;line</i>	Specifies the line to which the value will be assigned.
equal sign (=)	Indicates assignment.
<i>&amp;start</i>	Starting position. It can be: <ul style="list-style-type: none"><li>• An absolute number.</li><li>• A numeric field.</li><li>• The word "BEFORE" for placing the image before the line.</li></ul>



*&image* An image name as declared in the <object> section.

**Programming notes:**

- An image can be inserted before a line or on a line before text. Use the (BEFORE: 0) substring notation to place the image before a line.
- Images placed on a line before text push the line text to the right and may cause column alignment problems.

**Examples:**

This example places .imag1 on line 10 before position 15:

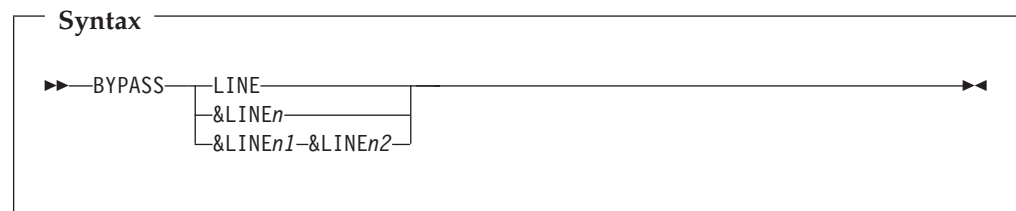
```
LINE10 (15: 0) = .imag1
```

This example places .imag1 before line 5:

```
LINE5 (BEFORE: 0) = .imag1
```

## BYPASS statement

The BYPASS statement tags specific lines or a range of lines for deletion. That is, the bypassed lines are excluded from the output document.



## Parameters

- LINE** The line number contained in the IDX subscript is bypassed.
- &LINE $n$**  Line number to bypass.
- &LINE $n1$**  Low line number of range of lines to bypass.
- &LINE $n2$**  High line number of range of lines to bypass.

## Examples

This example bypasses line 2:

```
IDX = 2
BYPASS LINE
```

This example bypasses line 4:

```
BYPASS LINE4
```

This example bypasses line 1 through line 10:

```
BYPASS LINE1-LINE10
```

## Programming notes

The BYPASS statement is convenient for stripping unneeded report lines from the output document, especially when creating CSV files. For example, running page titles and field headings are not needed in a spreadsheet. Report title lines and heading lines can be conditionally or unconditionally excluded from the output CSV file.

## BYPASS statement

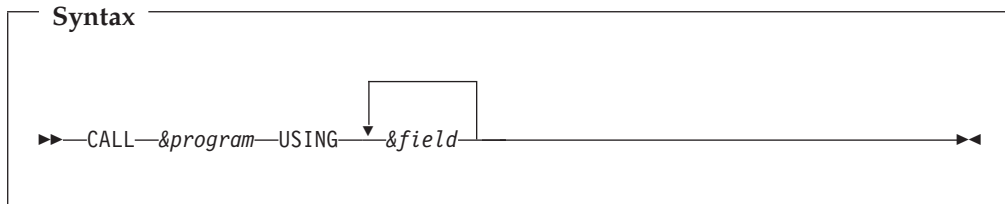
The use of BYPASS is not limited to CSV format files. It can also be used to bypass report lines when creating HTML and HTML1 format documents.

---

## CALL statement

The CALL statement makes a call to a user-written program. The user-written program can be written in COBOL, BAL, or any other language that honors standard linkage conventions.

### Syntax



### Parameters

*&program*

The program name. This can be:

- A hard-coded program name enclosed in quotes. In this case, it is a static call. That is, the program is included from the SYSLIB at link time.
- A field name that contains the program name. In this case, it is a dynamic call. The program must exist in the JOBLIB/STEPLIB at run time.

*&field*

Field name as declared in the Library section.

### Programming notes

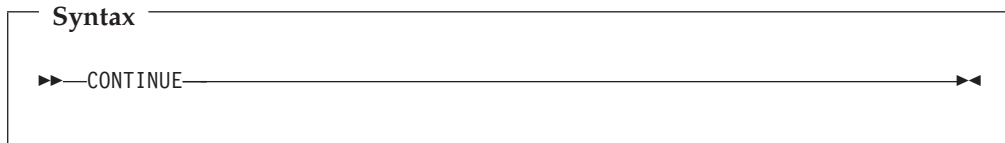
CALL *&program* USING *&field* must be coded on a single line. Additional fields can be coded on the same line or subsequent lines.

---

## CONTINUE statement

The CONTINUE statement is used in combination with the IF statement. It alters the processing logic to after the END-IF statement.

### Syntax



### Parameters

None.

## Examples

```
IF LINE1 (3: 4) = '1234'
  CONTINUE
ELSE
  processing statements
END-IF
```

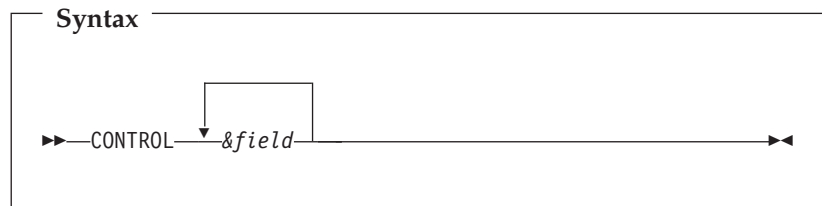
---

## CONTROL statement

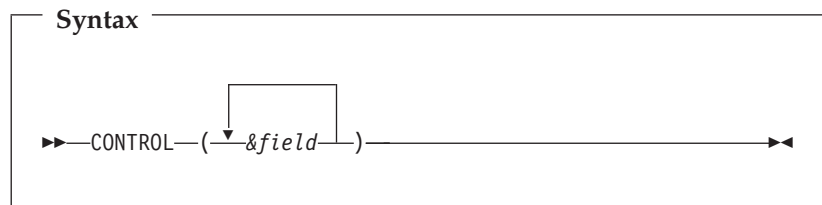
The CONTROL statement declares control break fields for an HTML1 format document. A selection tree is built on the left side of HTML1 format document in the hierarchy specified on the CONTROL statement.

There are two CONTROL statement formats:

**Format 1** Use this format when all fields can be listed on a single line.



**Format 2** Use this format when fields cannot be listed on a single line. The list begins with an open parenthesis and ends with a closed parenthesis. The list can span over multiple lines.



## Parameters

*&field* Field name as declared in the Library section.

## Programming notes

Reports normally consist of title lines, field headings, detail lines, and control break totals. That is, reports follow a hierarchy as enforced in the programs that create them.

To create a selection tree such that you can navigate directly to a specific page where a new control starts, you must first define fields and variables to hold control break information.

Then, in the Activity section, the fields are populated by control field values from specific positions in the page lines. When populating fields, care must be taken to use the correct position and length where a control break value is printed. This is usually achieved by testing for a specific literal or data type in specific locations on one or more lines of each page.

## CONTROL statement

Finally, control breaks are declared by the CONTROL statement. The first listed field is the highest break, followed by the subsequent fields as secondary breaks.

### Examples

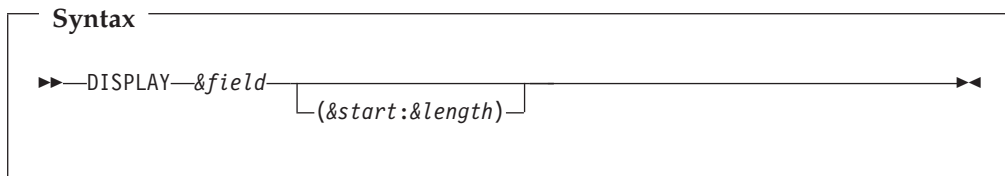
```
DEFINE COMPANY W 2 A
DEFINE BRANCH W 5 A
DEFINE OFFICER W 4 A
IF (LINE5 (2: 7) = 'COMPANY')
AND (LINE5 (13: 6) = 'BRANCH')
AND (LINE8 (4: 2) IS NUMERIC)
COMPANY = LINE8 (4: 2)
BRANCH = LINE8 (14: 3)
OFFICER = LINE8 (24: 4)
END-IF
CONTROL COMPANY BRANCH OFFICER
```

---

## DISPLAY statement

The DISPLAY statement prints fields, lines, or a literal to SYSOUT.

### Syntax



### Parameters

<i>&amp;field</i>	Field name, literal, or line.
<i>&amp;start</i>	Starting position.
<i>&amp;length</i>	Text length.

### Examples

```
DISPLAY 'COMPANY: ' COMPANY 'BRANCH: ' LINE1 (5: 4)
```

### Programming notes

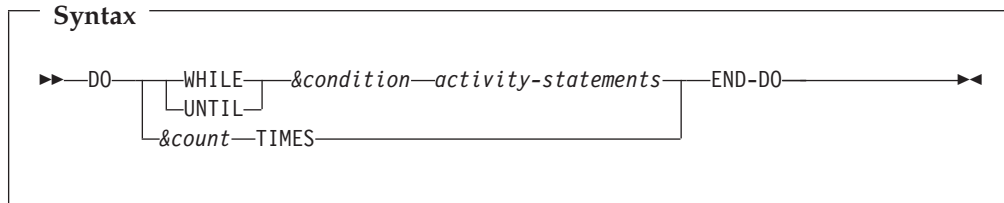
DISPLAY fields can be listed on multiple lines as needed. All packed decimal and binary fields are converted to numeric display format. The sign is not removed.

---

## DO and END-DO statements

The DO and END-DO statements define the scope of repetitive program logic.

## Syntax



## Parameters

<b>WHILE</b>	Evaluates the condition expression, <i>&amp;condition</i> , at the top of a group of statements.
<b>UNTIL</b>	Evaluates the condition expression, <i>&amp;condition</i> , at the bottom of a group of statements.
<i>&amp;condition</i>	Specifies the condition expression for the continuous execution of the loop. See “IF, ELSE, and END-IF statements” on page 47 for the conditional expression syntax.
<i>&amp;count</i>	The loop counter or limit. The maximum value is 2147483647. <i>&amp;count</i> must be a numeric field or a positive number.
<b>END-DO</b>	Terminates the DO statement.

## Programming notes

For DO WHILE, the truth value of the conditional expression, *&condition*, determines whether statements bound by the DO and END-DO pair are to be executed. When the conditional expression is true, the statements are executed. When the conditional expression is false, the processing continues with the next statement following the END-DO.

For DO UNTIL, the statements bound by the DO and END-DO pair are executed. The truth value of the conditional expression, *&condition* (evaluated at the end of the statements), determines whether statements bound by the DO and END-DO pair are to be executed again. When the conditional expression is true, the statements are executed again. When the conditional expression is false, the processing continues with the next statement following the END-DO.

For DO *&count* TIMES, the statements bound by the DO and END-DO pair are executed *&count* times unconditionally.

## Examples

```

DEFINE WS-COUNT W B 2 VALUE ZERO
DO UNTIL (WS-COUNT = 6)
    WS-COUNT= (WS-COUNT + 1)
    DISPLAY 'LOOP COUNT OF 6'
END-DO
DO 6 TIMES
    DISPLAY 'LOOP COUNT OF 6'
END-DO
  
```

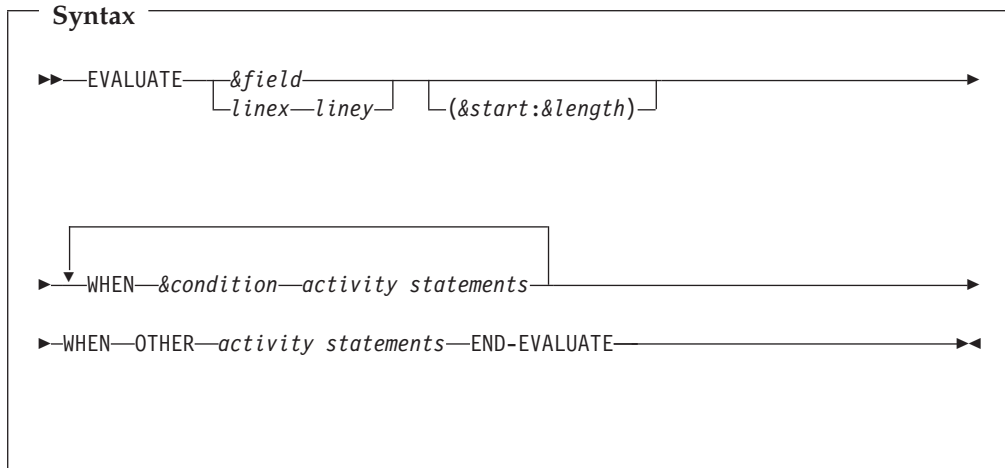
---

## EVALUATE and END-EVALUATE statements

The EVALUATE statement provides an elegant way of testing for values.

## EVALUATE and END-EVALUATE statements

### Syntax



### Parameters

<i>&amp;field</i>	The field name to be evaluated.
<i>linex-linex</i>	Range of lines to evaluate, where <i>linex</i> is the low line number and <i>linex</i> is the high line number. <i>linex</i> must be greater than zero and less or equal to <i>linex</i> . <i>linex</i> must be greater than zero and less or equal to LINE-COUNT.
<i>&amp;start</i>	Starting position.
<i>&amp;length</i>	Text length.
<i>&amp;condition</i>	Value to be tested for. It must be a literal or a field with the same data type as <i>&amp;field</i> .
<b>OTHER</b>	Must be the last statement after a series of tests. The statements following OTHER are executed only when all previous tests fail.
<b>END-EVALUATE</b>	Terminates the EVALUATE statement.

### Programming notes

When a line range is specified, the reserved field *IDX* is initialized to *linex* and EVALUATE statements are processed in a loop incrementing *IDX* by 1 until all lines in the specified range are tested. Up to 8 nested EVALUATE statements are permitted.

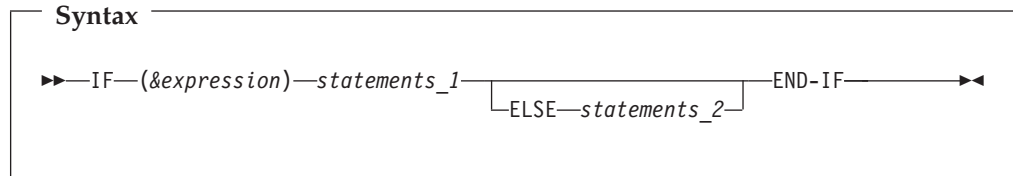
### Examples

```
EVALUATE LINE8-LINE55 (24: 4)
  WHEN 'DDDD'
    LINE (24: 4) = .EM1
  WHEN 'EBEE'
    LINE (24: 4) = .EM2
  WHEN OTHER
    LINE (24: 4) = .EM3
END-EVALUATE
```

## IF, ELSE, and END-IF statements

The IF statement conditionally controls execution of the statements bound by the IF and END-IF statements.

### Syntax



### Parameters

<i>&amp;expression</i>	Conditional expression. It can contain AND, OR, and arithmetic terms.
<i>statements_1</i>	The statements executed if <i>&amp;expression</i> is evaluated to be true.
<i>statements_2</i>	The statements executed if <i>&amp;expression</i> is evaluated to be false. If ELSE is not specified, then no statements are executed.
<b>END-IF</b>	Terminates the logic associated with the previous IF statement.

### Programming notes

*&expression* must be enclosed in parentheses. Multiple expressions can be coded, each starting on a separate line and connected with an AND or an OR logical operator.

### Examples

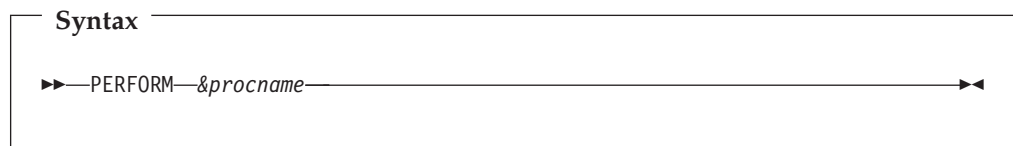
```

IF (LINE5 (2: 7) = 'COMPANY')
AND (LINE5 (13: 6) = 'BRANCH')
AND (LINE8 (4: 2) IS NUMERIC)
  COMPANY = LINE8 (4: 2)
  BRANCH = LINE8 (14: 3)
  EVALUATE LINE8-LINE55 (24: 4)
    WHEN 'DDDD'
      LINE (24: 4) = .em1
    WHEN 'EBEE'
      LINE (24: 4) = .em2
  END-EVALUATE
END-IF
  
```

## PERFORM statement

The PERFORM statement performs the specified procedure.

### Syntax



## PERFORM statement

### Parameters

*&procname* Procedure name 1 to 30 characters long. The name can consist of letters, numbers and hyphens. The procedure must be declared at the bottom of the program.

### Programming notes

For additional information, see "PROC/END-PROC statement."

---

## PROC/END-PROC statement

The PROC and END-PROC statements are used to declare the beginning and the end of a procedure in the Activity section.

### Syntax

```
Syntax
▶—&procname.—PROC—activity statements—END-PROC—▶
```

### Parameters

*&procname* Procedure name 1 to 30 characters long. The name can consist of letters, numbers and hyphens. The first character and the last character must be a letter or a number.

### Programming notes

Procedures are declared at the bottom of the script program. One or more procedures can be coded.

Procedures are invoked by means of the PERFORM statement as needed.

Procedures should be used for repetitive logic to make the script program manageable.

RMU does not support the GOTO statement. Therefore procedures must be written in a straight top-to-bottom design.

### Examples

```
PERFORM A0001-DECORATE

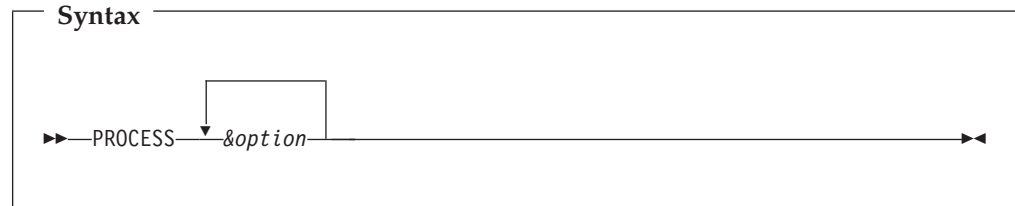
A0001-DECORATE. PROC
  IF (LINE5 (2: 7) = 'COMPANY')
  AND (LINE5 (13: 6) = 'BRANCH')
  AND (LINE8 (4: 2) IS NUMERIC)
    EVALUATE LINE8-LINE55 (24: 4)
      WHEN 'DDDD'
        LINE (24: 4) = .em1
      WHEN 'EBEE'
        LINE (24: 4) = .em2
    END-EVALUATE
  END-IF
END-PROC
```



## PROCESS statement

The PROCESS statement controls COBOL compiler options.

### Syntax



### Parameters

*&option* COBOL compiler PROCESS (CBL) option.

### Programming notes

PROCESS statements are placed before the PARM statement at the top of the program. Options are separated by a comma (.). Options can be coded up to position 64, but multiple PROCESS statements can be coded as needed.

Options are not validated by RMU. Incorrect options will cause the COBOL compiler to fail.

Refer to the *IBM COBOL Reference* manual for valid options.

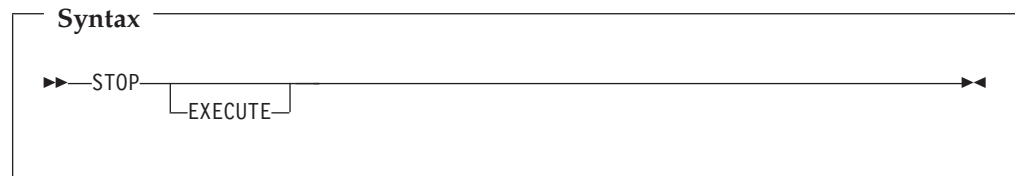
### Examples

```
PROCESS LIST,OPTIMIZE
PROCESS OUTDD(SYSPRINT)
```

## STOP statement

The STOP statement terminates the job.

### Syntax



### Programming notes

STOP terminates processing with RETURN-CODE set by the RMU Script logic.

STOP EXECUTE terminates processing with RETURN-CODE set to 16, unless RETURN-CODE was set to a non-zero value before the STOP EXECUTE was issued.



```

STRING WVALUE1      DELIMITED BY SIZE
      'UTILITY '    DELIMITED BY SIZE
      LINE8 (4: 3) DELIMITED BY ' '
      'BILLS'       DELIMITED BY SIZE
INTO WSTRING
END-STRING

```

At execution, WSTRING contains: "THIS PAYS UTILITY 10 BILLS".

**Case 2:**

```

WSTRING = 'PAYMENT INFORMATION: '
WCOUNT = 22
STRING WVALUE1      DELIMITED BY SIZE
      'UTILITY '    DELIMITED BY SIZE
      LINE8 (4: 3) DELIMITED BY ' '
      'BILLS'       DELIMITED BY SIZE
INTO WSTRING WITH POINTER WCOUNT
END-STRING

```

At execution, WSTRING contains: "PAYMENT INFORMATION: THIS PAYS UTILITY 10 BILLS"

## System-defined fields

Fields available to programmers:

**IDX** A 4-byte binary integer used as a subscript when LINE is coded in an argument without a substring.

Also, IDX is used as a subscript in the EVALUATE statement when evaluating a range of lines for a value.

**I** The same as IDX.

**IDX2** A 4-byte binary integer used as a work subscript for the BYPASS statement when bypassing a range of lines. This field is for RMU internal use only.

**I2** The same as IDX2.

**LINE** General purpose name for accessing lines within a page. When coded, the IDX subscript must contain a valid line number within the page buffer.

**LINE-COUNT** Contains the number of lines in the current page.

**LINE1** thru **LINE66** References specific lines within a page.

**PAGE-COUNT** A 4-byte binary integer containing the current page being processed. This field should not be confused with the actual page number that may be shown on report titles.

**RETURN-CODE** A 4-byte binary integer used as the job return code.

**SYSDATE** or **SYSDATE-E** An 8-byte alpha field representing the current date as YY/MM/DD.

**SYSDATE-9** A 6-byte numeric field representing the current date as YYMMDD.

## System-defined fields

### SYSDATE-LONG or SYSDATE-LONG-E

A 10-byte alpha field representing the current date as CCYY/MM/DD.

### SYSDATE-LONG-9

An 8-byte numeric field representing the current date as CCYYMMDD.

### SYSTIME

An 8-byte alpha field representing the current time as HH.MM.SS.

### SYSTIME-9

A 6-byte numeric field representing the current time as HHMMSS.

---

## COBOL verbs, statements, and reserved fields

These words cannot be used as field names in RMU Script:

ABEND-MESSAGE	B-NOT
ABEND-REQUEST	B-OR
ACCEPT	BACKGROUND-COLOR
ACCESS	BACKGROUND-COLOUR
ACCUM	BACKWARD
ACQUIRE	BASIS
ADD	BEEP
ADDRESS	BEFORE
ADVANCING	BEGINNING
AFTER	BELL
ALL	BINARY
ALLOWING	BIT
ALPHABET	BITS
ALPHABETIC-HIGHER	BLANK
ALPHABETIC-LOWER	BLINK
ALPHABETIC	BLOCK
ALPHANUMERIC-EDITED	BOOLEAN
ALPHANUMERIC	BOTTOM
ALSO	BY
ALTER	CALL
ALTERNATE	CANCEL
AND	CBL
ANY	CD
APPLY	CF
ARE	CH
AREA-VALUE	CHAIN
AREA	CHAINING
AREAS	CHANGED
ARITHMETIC	CHARACTER
ASCENDING	CHARACTERS
ASSIGN	CH1
AT	CH10
AUTHOR	CH11
AUTO-SKIP	CH12
AUTO	CH2
AUTOMATIC	CH3
B-AND	CH4
B-EXOR	CH5
B-LESS	CH6

## COBOL verbs, statements, and reserved fields

CH7  
CH8  
CH9  
CLASS  
CLOCK-UNITS  
CLOSE  
COBOL  
CODE-SET  
CODE  
COL  
COLLATING  
COLOR  
COLUMN  
COM-REG  
COMMA  
COMMAND-LINE  
COMMIT  
COMMITMENT  
COMMON  
COMMUNICATION  
COMP-X  
COMP-0  
COMP-1  
COMP-2  
COMP-3  
COMP-4  
COMP-5  
COMP-6  
COMP-7  
COMP-8  
COMP-9  
COMP  
COMPUTATIONAL-X  
COMPUTATIONAL-0  
COMPUTATIONAL-1  
COMPUTATIONAL-2  
COMPUTATIONAL-3  
COMPUTATIONAL-4  
COMPUTATIONAL-5  
COMPUTATIONAL-6  
COMPUTATIONAL-7  
COMPUTATIONAL-8  
COMPUTATIONAL-9  
COMPUTATIONAL  
COMPUTE  
CON  
CONFIGURATION  
CONNECT  
CONSOLE  
CONTAINED  
CONTAINS  
CONTENT  
CONTINUE  
CONTROL-AREA  
CONTROL  
CONTROLS  
CONVERTING  
COPY  
CORR  
CORRESPONDING  
COUNT  
CREATE  
CRT-UNDER  
CRT  
CTLFOOT  
CURRENCY  
CURRENT  
CURSOR  
CYCLE  
DATA  
DATE-COMPILED  
DATE-WRITTEN  
DATE  
DAY-OF-WEEK  
DAY  
DB-ACCESS-CONTROL-KEY  
DB-DATA-NAME  
DB-EXCEPTION  
DB-FORMAT-NAME  
DB-RECORD-NAME  
DB-SET-NAME  
DB-STATUS  
DB  
DBCS  
DE  
DEBUG-CONTENTS  
DEBUG-ITEM  
DEBUG-LINE  
DEBUG-NAME  
DEBUG-SUB-1  
DEBUG-SUB-2  
DEBUG-SUB3  
DEBUGGING  
DECIMAL-POINT  
DECLARATIVES  
DEFAULT  
DELETE  
DELIMITED  
DELIMITER  
DEPENDING  
DESCENDING  
DESTINATION  
DETAIL  
DISABLE  
DISCONNECT  
DISK  
DISPLAY-1  
DISPLAY-2  
DISPLAY-3  
DISPLAY-4  
DISPLAY-5  
DISPLAY-6

## COBOL verbs, statements, and reserved fields

DISPLAY-7	ERROR
DISPLAY-8	ESCAPE
DISPLAY-9	ESI
DISPLAY	EVALUATE
DIVIDE	EVERY
DIVISION	EXACT
DOWN	EXCEEDS
DROP	EXCEPTION
DUPLICATE	EXCESS-3
DUPLICATES	EXCLUSIVE
DYNAMIC	EXEC
EGCS	EXECUTE
EGI	EXHIBIT
EJECT	EXIT
ELSE	EXTEND
EMI	EXTERNAL
EMPTY-CHECK	EXTERNALLY-DESCRIBED-KEY
EMPTY	FALSE
ENABLE	FD
END-ACCEPT	FETCH
END-ADD	FILE-CONTROL
END-CALL	FILE-ID
END-COMPUTE	FILE
END-DELETE	FILLER
END-DISABLE	FINAL
END-DIVIDE	FIND
END-ENABLE	FINISH
END-EVALUATE	FIRST
END-IF	FIXED
END-MULTIPLY	FOOTING
END-OF-PAGE	FOR
END-PERFORM	FOREGROUND-COLOR
END-READ	FOREGROUND-COLOUR
END-RECEIVE	FORM
END-RECORD	FORMAT
END-RETURN	FREE
END-REWRITE	FROM
END-SEARCH	FULL
END-SEND	FUNCTION
END-START	GENERATE
END-STRING	GET
END-SUBTRACT	GIVING
END-TRANSCEIVE	GLOBAL
END-UNSTRING	GO
END-WRITE	GOBACK
END	GREATER
ENDG	GROUP
ENDING	HDR
ENTER	HEADER
ENTRY	HIGH-VALUE
ENVIRONMENT	HIGH-VALUES
EOP	HIGHLIGHT
EQ	I-O-CONTROL
EQUAL	I-O
EQUALS	ID
ERASE	IDENTIFICATION

## COBOL verbs, statements, and reserved fields

IDW01	LESS
IDW02	LIKE
IDW03	LIMIT
IDW04	LIMITS
IDW05	LINAGE-COUNTER
IDW06	LINAGE
IDW07	LINE-COUNTER
IDW08	LINES
IF	LINKAGE
IN	LIT
INDEX-1	LOCALLY
INDEX-2	LOCK
INDEX-3	LOW-VALUE
INDEX-4	LOW-VALUES
INDEX-5	MANUAL
INDEX-6	MASK
INDEX-7	MEMBER
INDEX-8	MEMORY
INDEX-9	MERGE
INDEX	MESSAGE
INDEXED	MODE
INDIC	MODIFIED
INDICATE	MODIFY
INDICATOR	MODULES
INDICATORS	MORE-LABELS
INITIAL	MOVE
INITIALIZE	MULTIPLE
INITIATE	MULTIPLY
INPUT-OUTPUT	NATIVE
INPUT	NEGATIVE
INSERT	NEXT
INSPECT	NO-ECHO
INSTALLATION	NO
INTO	NONE
INVALID	NORMAL
IS	NOT
IW-FAKE-INDEX	NULL
IW-PENGI-PAGE	NULLS
IW-RETURN-CODE	NUMBER
IW-WORK-INDEX	NUMERIC-EDITED
JAPANESE	NUMERIC
JUST	OBJECT-COMPUTER
JUSTIFIED	OBJECT
KANJI	OBJECT
KEEP	OCCURS
KEPT	OF
KEY	OFF
KEYBOARD	OMITTED
LAST	ON
LBEL	ONLY
LD	OPEN
LEADING	OPTIONAL
LEFT-JUSTIFY	OR
LEFT	ORDER
LENGTH-CHECK	ORGANIZATION
LENGTH	OTHER

## COBOL verbs, statements, and reserved fields

OUTPUT	RC-ABRCODE
OVERFLOW	RC-BOOLEAN
OWNER	RC-CHKOVF0
PACKED-DECIMAL	RC-COBSTAT
PADDING	RC-DATESWP
PAGE-COUNTER	RC-HEXSTR0
PAGE	RC-HEXSTR1
PAGEFOOT	RC-REPCHR0
PALETTE	RC
PARAGRAPH	RD
PARAM-LENGTH	READ
PARAM-REGISTER-9	READY
PARAM-REGISTER	REALM
PASSWORD	RECEIVE
PENGI-CODES	RECONNECT
PENGI-CODE0	RECORD-NAME
PENGI-STATUS	RECORD
PERFORM	RECORDING
PF	RECORDS
PH	REDEFINES
PIC	REEL
PICTURE	REFERENCE
PLUS	REFERENCES
POINTER	RELATION
POS	RELATIVE
POSITION	RELEASE
POSITIVE	RELOAD
PRESENT	REMAINDER
PREVIOUS	REMOVAL
PRINT-SWITCH	RENAMES
PRINTER-1	REPEATED
PRINTER	REPLACE
PRINTING	REPLACING
PRIOR	REPORT
PROCEDURE	REPORTING
PROCEDURES	REPORTS
PROCEED	REQUIRED
PROCESS	RERUN
PROCESSING	RESERVE
PROGRAM-DATE	RESET
PROGRAM-ID	RETAINING
PROGRAM-INFO-TABLE	RETRIEVAL
PROGRAM-NAME	RETURN
PROGRAM-OPSYS	REVERSE-VIDEO
PROGRAM-TIME	REVERSED
PROGRAM-TYPE	REWIND
PROGRAM	REWRITE
PROMPT	RF
PROTECTED	RH
PURGE	RIGHT-JUSTIFY
QUEUE	RIGHT
QUOTE	ROLLBACK
QUOTES	ROLLING
RANDOM	ROUNDED
RANGE	RUN
RC-ABEND00	SAME



## COBOL verbs, statements, and reserved fields

SCREEN	SUBFILE
SD	SUBPROGRAM
SEARCH	SUBTRACT
SECTION	SUM
SECURE	SUPPRESS
SECURITY	SWITCH-1
SEGMENT-LIMIT	SWITCH-2
SEGMENT	SWITCH-3
SEL	SWITCH-4
SELECT	SWITCH-5
SEND	SWITCH-6
SENTENCE	SWITCH-7
SEPARATE	SWITCH-8
SEQUENCE	SWITCH
SEQUENTIAL	SYMBOLIC
SERVICE	SYNC
SESSION-ID	SYNCHRONIZED
SET	TABLE
SHARED	TALLY
SHIFT-IN	TALLYING
SHIFT-OUT	TAPE
SIGN	TENANT
SIZE	TERMINAL
SKIP1	TERMINATE
SKIP2	TEST
SKIP3	TEXT
SORT-CONTROL	THAN
SORT-CORE-SIZE	THEN
SORT-FILE-SIZE	THROUGH
SORT-MERGE	THRU
SORT-MESSAGE	TIME
SORT-MODE-SIZE	TIMEOUT
SORT-RETURN	TIMES
SORT	TITLE
SOURCE-COMPUTER	TO
SOURCE	TOP
SPACE-FILL	TRACE
SPACE	TRAILING-SIGN
SPACES	TRAILING
SPECIAL-NAMES	TRANSACTION
STANDARD-1	TRANSCEIVE
STANDARD-2	TRUE
STANDARD-3	TYPE
STANDARD-4	UNDERLINE
STANDARD	UNEQUAL
START	UNIT
STARTING	UNLOCK
STATUS	UNSTRING
STOP	UNTIL
STORE	UP
STRING	UPDATE
STYLE	UPON
SUB-QUEUE-1	USAGE-MODE
SUB-QUEUE-2	USAGE
SUB-QUEUE-3	USE
SUB-SCHEMA	USER

## COBOL verbs, statements, and reserved fields

USING  
VAL  
VALID  
VALIDATE  
VALUE  
VALUES  
VARCHAR  
VARIABLE  
VARYING  
WAIT  
WHEN-COMPILED  
WHEN  
WITH  
WITHIN  
WORDS  
WORKING-STORAGE  
WRITE-ONLY  
WRITE  
ZERO-FILL  
ZERO  
ZEROES  
ZEROS

---

## Chapter 6. Messages

There are three types of messages that you can encounter when using RMU:

### COBOL compiler-generated messages

RMU converts Script programs to COBOL and then compiles the generated COBOL with a standard LE or z/OS COBOL compiler. COBOL 'I' and 'W' level errors are acceptable. 'E' level errors are serious errors and should be reported to IBM for correction as they can be due to RMU compiler-generated statements.

These messages are generated by the COBOL compiler and are beyond the scope of this document.

### RMU runtime error messages

Runtime messages are self-explanatory. File open errors are common due to improper or missing DD names.

For an explanation of runtime messages, refer to *z/OS MVS System Messages*.

### RMU compiler-generated messages (listed in this chapter)

RMU compiler-generated messages consist of three parts: A 7-digit message number followed by a comma, a 2-digit condition code and the message text. A condition code of 00 is an advisory message. A condition code of 12 is an error.

---

## RMU compiler-generated messages

---

### RMU0-01,00 AUTO COPY OF IMAGES IGNORED.

**Explanation:** <object> section was declared for HTML format document. This is an MNOTE statement. MNOTE is informational only.

**User response:** The images declared (if any) in the section will not be automatically included in the document.

This means that all images used in the document must exist in the &home/images directory on the server where the document is downloaded.

---

### RMU0-01,12 UNPAIRED PARENTHESES ON STATEMENT

**Explanation:** String was coded with unpaired parentheses.

**User response:** Code string inside a paired parentheses, example ( string ).

---

### RMU0-01,12 UNPAIRED QUOTES ON TEXT STATEMENT

**Explanation:** A quoted string is missing end quote.

**User response:** Add the quote as needed.

---

### RMU0-02,12 STATEMENTS ARE CODED BEYOND COL 71

**Explanation:** Text was found beyond column 71.

**User response:** Correct the problem.

---

### RMU0-03,12 UNPAIRED END-IF/END-DO/EVALUATE STATEMENT

**Explanation:** A scope terminator is missing. For IF statement, there is no matching END-IF. For DO statement there is no matching END-DO. For EVALUATE statement there is no matching END-EVALUATE.

**User response:** Add the required scope terminator.

---

### RMU0-04,12 "ELSE" IS OUT OF SEQUENCE

**Explanation:** ELSE was placed on inappropriate line.

**User response:** Correct the problem.

---

### RMU0-05,12 "CONTINUE" IS OUT OF SEQUENCE

**Explanation:** CONTINUE statement is out of sequence.

**User response:** Correct the problem.

## RMU compiler-generated messages

---

### RMU0-06,12 INCOMPLETE STATEMENT

**Explanation:** Statement requires more arguments.

**User response:** Complete the statement as required.

---

### RMU0-07,12 LINE IS ILLEGAL (USE LINE1-LINE66)

**Explanation:** The variable name is not allowed. It conflicts with the reserved LINE $n$  names.

**User response:** Use the correct name.

---

### RMU0-08,12 TARGET FIELD NAME EXCEEDS 16 CHRS

**Explanation:** The field name is longer than 16 characters.

**User response:** Reduce the field name down to 16 characters or less.

---

### RMU0-09,12 UNDEFINED VARIABLE

**Explanation:** Variable is not defined and it is not a reserved field.

**User response:** Correct the problem.

---

### RMU0-10,12 ILLEGAL NAME

**Explanation:** The name is illegal as coded.

**User response:** Refer to DEFINE statement for variable naming rules.

---

### RMU0-11,12 ILLEGAL SUBSTRING EXPRESSION

**Explanation:** The substring is illegal as coded.

**User response:** Correct the erroneous substring.

---

### RMU0-12,12 IF ARGUMENTS ARE REQUIRED

**Explanation:** IF statement is not followed by arguments.

**User response:** Complete the IF statement.

---

### RMU0-13,12 ARGS NOT ENCLOSED IN PARENTHESES

**Explanation:** IF arguments are coded without parentheses.

**User response:** IF arguments must be enclosed in parentheses.

---

### RMU0-14,12 SUB START+LENGTH EXCEEDS MAXIMUM

**Explanation:** The start position plus the length exceeds field capacity.

---

**User response:** Correct the substring values such that the sum of start position plus the length does not exceed the field length.

---

### RMU0-15,12 DUPLICATE FIELD NAME

**Explanation:** The field name has been previously defined or it conflicts with a reserved field name.

**User response:** Use a different name.

---

### RMU0-16,12 INVALID LENGTH OR EXCEEDS MAXIMUM

**Explanation:** The field length is invalid or it exceeds maximum allowed length.

**User response:** Correct the length as follows: Binary field can be 2 or 4 bytes Packed decimal field can be up to 9 bytes Display numeric field can be up to 18 bytes. Alpha field can be up to 32,767 bytes.

---

### RMU0-17,12 TYPE NOT A, N, B OR P

**Explanation:** Unknown field type.

**User response:** Code the correct type.

---

### RMU0-18,12 STATEMENT CONTAINS EXTRA ARGUMENTS

**Explanation:** Extraneous arguments have been detected.

**User response:** Remove the extraneous arguments.

---

### RMU0-19,12 VALUE IS ILLEGAL AS WRITTEN

**Explanation:** Field value cannot be resolved as written.

**User response:** Correct the syntax.

---

### RMU0-20,12 VALUE EXCEEDS FIELD CAPACITY

**Explanation:** The specified value exceeds field capacity.

**User response:** Reduce the value to fit the field size.

---

### RMU0-21,12 STATEMENT IS OUT OF SEQUENCE

**Explanation:** Statement is out of sequence.

**User response:** Put statements in the correct order according to RMU syntax rules.

---

### RMU0-22,12 &EXPRESS ILLEGAL EXPRESSION

**Explanation:** Expression is illegal as written.

**User response:** Correct erroneous expression.

---

---

**RMU0-23,12 EXCEEDS 2048 NESTED IF STATEMENTS**

**Explanation:** The maximum number of nested IF statements has been exceeded.

**User response:** Reduce the number of IF statements. Simplify the tests.

---

**RMU0-24,12 NAME IS INVALID OR EXCEEDS 16 CHRS**

**Explanation:** The field name is longer than 16 characters.

**User response:** Reduce the field name down to 16 characters or less.

---

**RMU0-25,12 &parm INVALID OPTION**

**Explanation:** &parm is unknown option.

**User response:** Use the correct option.

---

**RMU0-26,12 &var VARIABLE IS NOT NUMERIC TYPE**

**Explanation:** Non-numeric variable used in substring.

**User response:** Use a numeric value or a numeric variable.

---

**RMU0-27,12 INVALID "DO" STATEMENT SYNTAX**

**Explanation:** The DO statement is illegal as written.

**User response:** Correct the statement according to DO statement rules.

---

**RMU0-28,12 "TIMES" NOT FOUND**

**Explanation:** Incomplete DO &nn statement.

**User response:** Correct the statement according to DO &nn TIMES rules.

---

**RMU0-29,12 "W" IS MISSING IN DEFINE STATEMENT**

**Explanation:** DEFINE is improper. "W" following the field name is required.

**User response:** Correct the erroneous statement.

---

**RMU0-30,12 &var INCOMPATIBLE VARIABLE TYPE**

**Explanation:** The &var type is not compatible with the IF argument type.

**User response:** Correct the erroneous argument.

---

**RMU0-30,12 &var INCOMPATIBLE VARIABLE TYPE**

**Explanation:** The &var type is not compatible with the WHEN argument type.

**User response:** Correct the erroneous argument.

---

**RMU0-31,12 "LINE" CANNOT BE A TARGET FIELD**

**Explanation:** LINE is improperly used as a target in ASSIGN statement.

**User response:** Correct the statement. LINE can be a target of an object or a style only. LINE content cannot be altered by RMU.

---

**RMU0-32,12 LITERAL EXCEEDS 60 CHARACTERS**

**Explanation:** Literal is too long.

**User response:** Reduce the literal size.

---

**RMU0-33,12 INVALID ASSIGN STATEMENT SYNTAX**

**Explanation:** Assign statement is not valid as written.

**User response:** Correct the statement to comply with the ASSIGN statement rules.

---

**RMU0-34,12 SUBSTRING NOT VALID FOR P,B TYPE**

**Explanation:** A substring was coded for a Packed decimal or a Binary field.

**User response:** Remove the substring.

---

**RMU0-35,12 RECURSIVE USE OF "&word"**

**Explanation:** The &word section was previously declared.

**User response:** Remove the extraneous section.

---

**RMU0-36,12 &word : ILLEGAL NUMBER/ ARGUMENT**

**Explanation:** Argument is illegal as written.

**User response:** Write the correct argument.

---

**RMU0-37,12 NAME EXCEEDS 8 CHARACTERS**

**Explanation:** Style or Object name exceeds 8 characters.

**User response:** Reduce the name down to 8 characters. Note that period is included as part of the name.

## RMU compiler-generated messages

---

### RMU0-38,12 ARGUMENT IS OUT OF SEQUENCE

**Explanation:** The argument is out of sequence.

**User response:** Style and Object arguments are written in pairs as `&name {...}`. Correct the erroneous arguments.

---

### RMU0-39,12 OBJECT USE, SUBSTRING LENGTH NOT 0

**Explanation:** The length in substring is not zero.

**User response:** When an object is used as source in assign, the length in the target substring must be zero.

---

### RMU0-40,12 &name DUPLICATE OBJECT NAME

**Explanation:** The `&name` was previously declared.

**User response:** Use a unique name.

---

### RMU0-41,12 &name UNDEFINED OBJECT NAME

**Explanation:** The `&name` is undefined.

**User response:** Use a defined `&name` from the Style or Object section.

---

### RMU0-42,12 "TURF" IS ILLEGAL IN THIS CONTEXT

**Explanation:** Illegal use of `.turf1` or `.turf2`.

**User response:** Correct the erroneous statement.

---

### RMU0-43,12 ZERO START IN SUBSTRING IS ILLEGAL

**Explanation:** Zero value was used as start position in target field of Assign statement.

**User response:** Correct the erroneous start value.

---

### RMU0-44,12 SYNTAX UNSUPPORTED FOR CSV FORMAT

**Explanation:** An object or a style name was used as source in Assign statement for CSV document.

**User response:** CSV documents cannot be decorated with fonts and images. Remove the statement.

---

### RMU0-45,12 </&tag> TAG IS MISSING

**Explanation:** No matching end tag was located for `<&tag>` section.

**User response:** Insert the correct `</&tag>` to complete the `<&tag>` section.

---

### RMU0-46,12 LITERAL FOUND IN CALL PARAMETERS

**Explanation:** Alphanumeric (quoted) or numeric literal was detected in the call parameters. Literal is not allowed in the CALL parameters.

**User response:** Remove erroneous parameter. Define a field in the library section with appropriate value and use the field name in the parameter list.

---

### RMU0-47,12 PROCESS STRING EXCEEDS 64 CHARS

**Explanation:** Parameters on the PROCESS statement span beyond column 64.

**User response:** Limit PROCESS statement to less than 65 characters. Code multiple PROCESS statements back to back to accommodate your options.

---

### RMU0-42,12 "TURF" IS ILLEGAL IN THIS CONTEXT

**Explanation:** `.turf` is being assigned to a LINE text.

**User response:** `.turf` is a special style that can be declared on the PARM statement. When coded, the option is applied to the entire document. Remove/correct improper statement.

---

### RMU0-43,12 ZERO START IN SUBSTRING IS ILLEGAL

**Explanation:** Zero was used for substring start position.

**User response:** Code the correct start position.

---

### RMU0-44,12 SYNTAX UNSUPPORTED FOR CSV FORMAT

**Explanation:** A style or and object was coded in assign statement.

**User response:** Style and objects cannot be assigned to text in CSV files. Remove erroneous statements.

---

### RMU0-45,12 </XXXX-XXX> TAG IS MISSING

**Explanation:** A matching end tag is not coded for the tag option in effect.

**User response:** Code the end tag in the appropriate place.

---

### RMU0-46,12 LITERAL FOUND IN CALL PARAMETERS

**Explanation:** Literal was coded on CALL statement.

**User response:** Define a field in Library section with the needed value and use the field name instead.

---

**RMU0-47,12 PROCESS STRING EXCEEDS 64 CHARS**

**Explanation:** COBOL PROCESS exceeds 64 characters.

**User response:** Code options using multiple PROCESS statement back to back.

---

**RMU0-48,12 EXCEEDS EIGHT(8) EVALUATE NESTS**

**Explanation:** Maximum nested EVALUATE statements exceeded.

**User response:** Reduce the number of nested EVALUATE statements. If you need to code more than 8 nested EVALUATE statements, write a procedure and use PERFORM to invoke the procedure name.

---

**RMU0-49,12 UNPAIRED END-EVALUATE STATEMENT**

**Explanation:** END-EVALUATE was not coded.

**User response:** Add END-EVALUATE to appropriate place.

---

**RMU0-50,12 DUPLICATE PROC NAME**

**Explanation:** Procedure name was previously defined.

**User response:** Use a unique procedure name.

---

**RMU0-51,12 "&proc" UNDEFINED PROC NAME**

**Explanation:** The "&proc" procedure name is not defined.

**User response:** Correct the spelling of your proc name.

---

**RMU0-52,12 INVALID PROC NAME**

**Explanation:** Procedure name contains invalid characters.

**User response:** Correct the name. You can use letters, numbers and hyphens. The first character and the last character must be a letter or a number.

---

**RMU0-53,12 END-PROC IS OUT OF SEQUENCE**

**Explanation:** END-PROC was placed without the PROC statement before it, or END-PROC was placed in the middle of a statement that requires a scope terminator, i.e., IF/DO or EVALUATE.

**User response:** Correct the problem.

---

**RMU0-54,12 PROC NAME EXCEEDS 30 CHARACTERS**

**Explanation:** Procedure name is more than 30 characters long.

**User response:** Reduce the name to 30 characters or less.

---

**RMU0-55,12 PROC PERFORM WOULD CAUSE A LOOP**

**Explanation:** perform &procname was placed in a proc where &procname is the procedure name where PERFORM was placed.

**User response:** Correct &procname or remove PERFORM statement in error.

---

**RMU0-56,12 END-PROC IS MISSING**

**Explanation:** End of script program is reached with a proc without a matching END-PROC statement.

**User response:** Add END-PROC as required.

---

**RMU0-57,12 ILLEGAL CHARACTER AFTER END-STRING**

**Explanation:** The END-STRING statement is followed by extraneous characters.

**User response:** Remove the extraneous characters.

---

**RMU0-58,12 "DELIMITED" OPTION IS MISSING**

**Explanation:** Improper STRING statement syntax.

**User response:** Refer to STRING statement for the correct syntax in this manual.





---

## Notices

This information was developed for products and services offered in the USA.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785  
USA

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation  
Licensing  
2-31 Roppongi 3-chome, Minato-ku  
Tokyo 106, Japan

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:**

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation  
Mail Station P300  
2455 South Road  
Poughkeepsie, NY 12601-5400  
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

---

## Trademarks

A current list of IBM trademarks is available on the Web at "Copyright and trademark information", <http://www.ibm.com/legal/copytrade.shtml>.

# Index

## Special characters

.ENUM1, parameter, assignment statement 40  
.ETEXT, parameter, assignment statement 40  
<docs\_end> statement 37  
<docs> statement 36  
<object> statement 35  
<page\_end> statement 38  
<page\_top> statement 37  
<style> statement 35  
&char, parameter  
    PARM statement 33  
    STRING statement 50  
&color, parameter, <style> statement 36  
&condition, parameter  
    DO statement 45  
    EVALUATE statement 46  
&count, parameter, DO statement 45  
&css parameter 3, 4  
&currency, parameter, PARM statement 33  
&decimal, parameter, PARM statement 33  
&expression, parameter, IF statement 47  
&field, parameter  
    CALL statement 42  
    CONTROL statement 43  
    DEFINE statement 34  
    DISPLAY statement 44  
    EVALUATE statement 46  
    STRING statement 50  
&format, parameter, PARM statement 33  
&formula, parameter, assignment statement 39  
&image, parameter  
    <object> statement 35  
    assignment statement 41  
&length, parameter  
    assignment statement 39, 40  
    DEFINE statement 34  
    DISPLAY statement 44  
    EVALUATE statement 46  
&line, parameter, assignment statement 40  
&LINEn, parameter, BYPASS statement 41  
&literal, parameter, STRING statement 50  
&mod, parameter, <object> statement 35  
&n, parameter, <style> statement 36  
&name, parameter  
    PARM statement 33  
&option, parameter, PROCESS statement 49  
&pointer, parameter, STRING statement 50  
&procname, parameter  
    PERFORM statement 48  
    PROC statement 48

&program, parameter, CALL statement 42  
&recfield, parameter, assignment statement 39  
&sendfield, parameter, assignment statement 39  
&size, parameter, <style> statement 36  
&start, parameter  
    assignment statement 39, 40  
    DISPLAY statement 44  
    EVALUATE statement 46  
&style, parameter, assignment statement 40  
&svc99, FJSVC99 file 7, 14  
&sys1 symbol 14  
&SYS1.SFZHAMAC library 11, 16  
&SYS1.SFZHASRC library 11, 14, 16  
&SYS1.SFZHDOCS library 7, 9, 11, 16, 30, 35  
&SYS1.SFZHJCLS library 3, 4, 5, 11, 12, 13, 14, 15, 18, 19, 21, 24, 26, 29  
&SYS1.SFZHLOAD library 11, 14  
&SYS1.SFZHPROC library 11, 15  
&target, parameter, STRING statement 50  
&type, parameter, DEFINE statement 34  
&value, parameter, DEFINE statement 34  
#BASE, sequence number type 30  
#EOF, sequence number type 31  
#nnnnn, sequence number type 30  
= (equal sign), assignment statement 39, 40

## Numerics

1403 paper 1, 19, 26, 30  
1403-paper option 3, 13, 36  
1403-paper parameter, PARM statement 36  
1403-paper, parameter, PARM statement 33

## A

A, parameter, DEFINE statement 34  
ABEND-MESSAGE  
    *See* RMU Script, invalid field names  
ABEND-REQUEST  
    *See* RMU Script, invalid field names  
ABENDs 31  
ACCEPT  
    *See* RMU Script, invalid field names  
ACCESS  
    *See* RMU Script, invalid field names  
ACCUM  
    *See* RMU Script, invalid field names  
ACQUIRE  
    *See* RMU Script, invalid field names

ADD  
    *See* RMU Script, invalid field names  
ADDRESS  
    *See* RMU Script, invalid field names  
ADVANCING  
    *See* RMU Script, invalid field names  
AFTER  
    *See* RMU Script, invalid field names  
ALL  
    *See* RMU Script, invalid field names  
ALLOWING  
    *See* RMU Script, invalid field names  
ALPHABET  
    *See* RMU Script, invalid field names  
ALPHABETIC  
    *See* RMU Script, invalid field names  
ALPHABETIC-HIGHER  
    *See* RMU Script, invalid field names  
ALPHABETIC-LOWER  
    *See* RMU Script, invalid field names  
ALPHANUMERIC  
    *See* RMU Script, invalid field names  
ALPHANUMERIC-EDITED  
    *See* RMU Script, invalid field names  
ALSO  
    *See* RMU Script, invalid field names  
ALTER  
    *See* RMU Script, invalid field names  
ALTERNATE  
    *See* RMU Script, invalid field names  
AND  
    *See* RMU Script, invalid field names  
ANY  
    *See* RMU Script, invalid field names  
APPLY  
    *See* RMU Script, invalid field names  
ARE  
    *See* RMU Script, invalid field names  
AREA  
    *See* RMU Script, invalid field names  
AREA-VALUE  
    *See* RMU Script, invalid field names  
AREAS  
    *See* RMU Script, invalid field names  
ARITHMETIC  
    *See* RMU Script, invalid field names  
ASCENDING  
    *See* RMU Script, invalid field names  
ASCII 4, 5, 7, 15, 16  
ASSIGN  
    *See* RMU Script, invalid field names  
assignment statement 38  
AT  
    *See* RMU Script, invalid field names  
AUTHOR  
    *See* RMU Script, invalid field names  
AUTO  
    *See* RMU Script, invalid field names  
AUTO-SKIP  
    *See* RMU Script, invalid field names

AUTOMATIC  
See RMU Script, invalid field names

## B

B-AND  
See RMU Script, invalid field names  
B-EXOR  
See RMU Script, invalid field names  
B-LESS  
See RMU Script, invalid field names  
B-NOT  
See RMU Script, invalid field names  
B-OR  
See RMU Script, invalid field names  
B, parameter, DEFINE statement 34  
background color  
defining 3, 30  
overriding 30  
BACKGROUND-COLOR  
See RMU Script, invalid field names  
BACKGROUND-COLOUR  
See RMU Script, invalid field names  
BACKWARD  
See RMU Script, invalid field names  
BASIS  
See RMU Script, invalid field names  
BEEP  
See RMU Script, invalid field names  
BEFORE  
See RMU Script, invalid field names  
BEGINNING  
See RMU Script, invalid field names  
BELL  
See RMU Script, invalid field names  
BINARY  
See RMU Script, invalid field names  
BIT  
See RMU Script, invalid field names  
BITS  
See RMU Script, invalid field names  
BLANK  
See RMU Script, invalid field names  
BLINK  
See RMU Script, invalid field names  
BLOCK  
See RMU Script, invalid field names  
BOOLEAN  
See RMU Script, invalid field names  
BOTTOM  
See RMU Script, invalid field names  
BY  
See RMU Script, invalid field names  
BYPASS statement 21, 24, 26, 41

## C

CALL  
See also RMU Script, invalid field names  
field name 52, 53  
statement 42  
CANCEL  
See RMU Script, invalid field names  
Cascaded Style Sheets  
See CSS

CBL  
See RMU Script, invalid field names  
CD  
See RMU Script, invalid field names  
CF  
See RMU Script, invalid field names  
CH  
See RMU Script, invalid field names  
CH1  
See RMU Script, invalid field names  
CH10  
See RMU Script, invalid field names  
CH11  
See RMU Script, invalid field names  
CH12  
See RMU Script, invalid field names  
CH2  
See RMU Script, invalid field names  
CH3  
See RMU Script, invalid field names  
CH4  
See RMU Script, invalid field names  
CH5  
See RMU Script, invalid field names  
CH6  
See RMU Script, invalid field names  
CH7  
See RMU Script, invalid field names  
CH8  
See RMU Script, invalid field names  
CH9  
See RMU Script, invalid field names  
CHAIN  
See RMU Script, invalid field names  
CHAINING  
See RMU Script, invalid field names  
CHANGED  
See RMU Script, invalid field names  
CHARACTER  
See RMU Script, invalid field names  
Character Separated Values  
See CSV  
CHARACTERS  
See RMU Script, invalid field names  
characters, control 18  
CLASS  
See RMU Script, invalid field names  
CLOCK-UNITS  
See RMU Script, invalid field names  
CLOSE  
See RMU Script, invalid field names  
COBLIB parameter 14  
COBLIST  
file, description 7  
file, required when compiling 5  
link and go mode 6  
COBOL  
See also RMU Script, invalid field names  
#BASE statement 30  
compiler  
generated messages 59  
options 49  
PROCESS (CBL) option 49  
requirement for 17  
currency symbol 13  
debugger 31

COBOL (continued)  
default  
compiler LIST option 12  
printer 14  
FZHRMUS1 utility 13  
listings, redirecting 14  
option, COBLIST file 7  
reserved  
fields 52  
words 16  
runtime problems 30, 31  
skeleton 16  
statement sequence numbers 30  
statements 52  
verbs 52  
CODE  
See RMU Script, invalid field names  
code set (ASCII or EBCDIC) 15  
CODE-SET  
See RMU Script, invalid field names  
coding, RMU Script 18  
COL  
See RMU Script, invalid field names  
COLLATING  
See RMU Script, invalid field names  
COLOR  
See RMU Script, invalid field names  
color, background 3  
COLUMN  
See RMU Script, invalid field names  
column values, formatting 21  
COM-REG  
See RMU Script, invalid field names  
COMMA  
See RMU Script, invalid field names  
COMMA, parameter, PARM  
statement 33  
COMMAND-LINE  
See RMU Script, invalid field names  
commands  
reading syntax diagrams xi  
COMMIT  
See RMU Script, invalid field names  
COMMITMENT  
See RMU Script, invalid field names  
COMMON  
See RMU Script, invalid field names  
COMMUNICATION  
See RMU Script, invalid field names  
COMP  
See RMU Script, invalid field names  
COMP-0  
See RMU Script, invalid field names  
COMP-1  
See RMU Script, invalid field names  
COMP-2  
See RMU Script, invalid field names  
COMP-3  
See RMU Script, invalid field names  
COMP-4  
See RMU Script, invalid field names  
COMP-5  
See RMU Script, invalid field names  
COMP-6  
See RMU Script, invalid field names  
COMP-7  
See RMU Script, invalid field names

COMP-8  
*See* RMU Script, invalid field names  
 COMP-9  
*See* RMU Script, invalid field names  
 COMP-X  
*See* RMU Script, invalid field names  
 COMPUTATIONAL  
*See* RMU Script, invalid field names  
 COMPUTATIONAL-0  
*See* RMU Script, invalid field names  
 COMPUTATIONAL-1  
*See* RMU Script, invalid field names  
 COMPUTATIONAL-2  
*See* RMU Script, invalid field names  
 COMPUTATIONAL-3  
*See* RMU Script, invalid field names  
 COMPUTATIONAL-4  
*See* RMU Script, invalid field names  
 COMPUTATIONAL-5  
*See* RMU Script, invalid field names  
 COMPUTATIONAL-6  
*See* RMU Script, invalid field names  
 COMPUTATIONAL-7  
*See* RMU Script, invalid field names  
 COMPUTATIONAL-8  
*See* RMU Script, invalid field names  
 COMPUTATIONAL-9  
*See* RMU Script, invalid field names  
 COMPUTATIONAL-X  
*See* RMU Script, invalid field names  
 COMPUTE  
*See* RMU Script, invalid field names  
 CON  
*See* RMU Script, invalid field names  
 CONFIGURATION  
*See* RMU Script, invalid field names  
 CONNECT  
*See* RMU Script, invalid field names  
 CONSOLE  
*See* RMU Script, invalid field names  
 constants  
 PROCLIB0 14  
 PRODUCT0 14  
 CONTAINED  
*See* RMU Script, invalid field names  
 CONTAINS  
*See* RMU Script, invalid field names  
 CONTENT  
*See* RMU Script, invalid field names  
 CONTINUE  
*See also* RMU Script, invalid field names  
 field name 53  
 statement 42  
 control  
 break  
 annotations, removing 21  
 fields, populating 26  
 selection tree 26  
 characters 18  
 CONTROL  
*See also* RMU Script, invalid field names  
 field name 53  
 statement 43  
 CONTROL statement 26

CONTROL-AREA  
*See* RMU Script, invalid field names  
 CONTROLS  
*See* RMU Script, invalid field names  
 CONVERTING  
*See* RMU Script, invalid field names  
 COPY  
*See* RMU Script, invalid field names  
 CORR  
*See* RMU Script, invalid field names  
 CORRESPONDING  
*See* RMU Script, invalid field names  
 COUNT  
*See* RMU Script, invalid field names  
 CREATE  
*See* RMU Script, invalid field names  
 CRT  
*See* RMU Script, invalid field names  
 CRT-UNDER  
*See* RMU Script, invalid field names  
 CSS  
 <style> tag 35  
 &css parameter 3  
 &SYS1.SFZHD0CS library 11  
 applying to CSV file 21  
 assignment statement 39  
 base templates 16  
 FJIDOC0 file 7  
 FZHBPARS parser utility 8  
 PARM statement 3  
 style section 30  
 tree templates 16  
 CSV  
 applying CSS, images 21  
 assignment statement 39  
 BYPASS statement 41  
 CSVCHAR parameter 13  
 default separator character 13  
 distributing output documents 4  
 document, creating 21  
 example, RMU script program 22  
 files 5  
 FJCONFG  
 file description 7  
 not in JCL 15  
 FJORPT0, file description 7  
 FZHCSV00, I/O module 16  
 introduction to RMU 1  
 parameter, PARM statement 33  
 CSVCHAR parameter 13  
 CTLFOOT  
*See* RMU Script, invalid field names  
 CURRENCY  
 parameter 11  
 currency symbol 13  
 CURRENT  
*See* RMU Script, invalid field names  
 CURSOR  
*See* RMU Script, invalid field names  
 custom jobs, publishing to z/OS  
 UNIX 15  
 CWORK parameter 14  
 CYCLE  
*See* RMU Script, invalid field names

## D

DATA  
*See* RMU Script, invalid field names  
 DATE  
*See* RMU Script, invalid field names  
 DATE-COMPILED  
*See* RMU Script, invalid field names  
 DATE-WRITTEN  
*See* RMU Script, invalid field names  
 DAY  
*See* RMU Script, invalid field names  
 DAY-OF-WEEK  
*See* RMU Script, invalid field names  
 DB  
*See* RMU Script, invalid field names  
 DB-ACCESS-CONTROL-KEY  
*See* RMU Script, invalid field names  
 DB-DATA-NAME  
*See* RMU Script, invalid field names  
 DB-EXCEPTION  
*See* RMU Script, invalid field names  
 DB-FORMAT-NAME  
*See* RMU Script, invalid field names  
 DB-RECORD-NAME  
*See* RMU Script, invalid field names  
 DB-SET-NAME  
*See* RMU Script, invalid field names  
 DB-STATUS  
*See* RMU Script, invalid field names  
 DBCS  
*See* RMU Script, invalid field names  
 DD names  
 FJCONFG 5, 15  
 FJSYSR 14  
 FJUNIX0 4, 14  
 summary 5  
 DE  
*See* RMU Script, invalid field names  
 DEBUG-CONTENTS  
*See* RMU Script, invalid field names  
 DEBUG-ITEM  
*See* RMU Script, invalid field names  
 DEBUG-LINE  
*See* RMU Script, invalid field names  
 DEBUG-NAME  
*See* RMU Script, invalid field names  
 DEBUG-SUB-1  
*See* RMU Script, invalid field names  
 DEBUG-SUB-2  
*See* RMU Script, invalid field names  
 DEBUG-SUB3  
*See* RMU Script, invalid field names  
 debugger, COBOL 31  
 DEBUGGING  
*See* RMU Script, invalid field names  
 debugging, RMU Script programs 30  
 DECIMAL parameter 11  
 DECIMAL-POINT  
*See* RMU Script, invalid field names  
 DECLARATIVES  
*See* RMU Script, invalid field names  
 decorating  
 file types 29  
 HTML document 29  
 HTML-format document 24  
 HTML1-format document 26

default  
  library locator table 13  
  options table 11

DEFAULT  
  *See* RMU Script, invalid field names

DEFINE statement 26, 34

DELETE  
  *See* RMU Script, invalid field names

DELIMITED  
  *See* RMU Script, invalid field names

DELIMITER  
  *See* RMU Script, invalid field names

DEPENDING  
  *See* RMU Script, invalid field names

DESCENDING  
  *See* RMU Script, invalid field names

DESTINATION  
  *See* RMU Script, invalid field names

DETAIL  
  *See* RMU Script, invalid field names

directories, UNIX 5

DISABLE  
  *See* RMU Script, invalid field names

DISCONNECT  
  *See* RMU Script, invalid field names

DISK  
  *See* RMU Script, invalid field names

DISPLAY  
  *See* RMU Script, invalid field names

DISPLAY statement 44

DISPLAY-1  
  *See* RMU Script, invalid field names

DISPLAY-2  
  *See* RMU Script, invalid field names

DISPLAY-3  
  *See* RMU Script, invalid field names

DISPLAY-4  
  *See* RMU Script, invalid field names

DISPLAY-5  
  *See* RMU Script, invalid field names

DISPLAY-6  
  *See* RMU Script, invalid field names

DISPLAY-7  
  *See* RMU Script, invalid field names

DISPLAY-8  
  *See* RMU Script, invalid field names

DISPLAY-9  
  *See* RMU Script, invalid field names

distributing, output documents 4

DIVIDE  
  *See* RMU Script, invalid field names

DIVISION  
  *See* RMU Script, invalid field names

DO  
  loop, controlling 21  
  statement 44

docs statement 36

docs\_end statement 37

DOWN  
  *See* RMU Script, invalid field names

downloading to a PC 5

DROP  
  *See* RMU Script, invalid field names

DUPLICATE  
  *See* RMU Script, invalid field names

DUPLICATES  
  *See* RMU Script, invalid field names

DYNAMIC  
  *See* RMU Script, invalid field names

**E**

EBCDIC 5, 15, 16

EGCS  
  *See* RMU Script, invalid field names

EGI  
  *See* RMU Script, invalid field names

EJECT  
  *See* RMU Script, invalid field names

ELSE  
  *See* RMU Script, invalid field names

ELSE statement 47

EMI  
  *See* RMU Script, invalid field names

EMPTY  
  *See* RMU Script, invalid field names

EMPTY-CHECK  
  *See* RMU Script, invalid field names

ENABLE  
  *See* RMU Script, invalid field names

END  
  *See* RMU Script, invalid field names

END-ACCEPT  
  *See* RMU Script, invalid field names

END-ADD  
  *See* RMU Script, invalid field names

END-CALL  
  *See* RMU Script, invalid field names

END-COMPUTE  
  *See* RMU Script, invalid field names

END-DELETE  
  *See* RMU Script, invalid field names

END-DISABLE  
  *See* RMU Script, invalid field names

END-DIVIDE  
  *See* RMU Script, invalid field names

END-DO statement 44

END-ENABLE  
  *See* RMU Script, invalid field names

END-EVALUATE  
  *See* RMU Script, invalid field names

END-EVALUATE statement 45

END-IF  
  *See also* RMU Script, invalid field names  
  field name 54  
  statement 47

END-MULTIPLY  
  *See* RMU Script, invalid field names

END-OF-PAGE  
  *See* RMU Script, invalid field names

END-PERFORM  
  *See* RMU Script, invalid field names

END-PROC statement 48

END-READ  
  *See* RMU Script, invalid field names

END-RECEIVE  
  *See* RMU Script, invalid field names

END-RECORD  
  *See* RMU Script, invalid field names

END-RETURN  
  *See* RMU Script, invalid field names

END-REWRITE  
  *See* RMU Script, invalid field names

END-SEARCH  
  *See* RMU Script, invalid field names

END-SEND  
  *See* RMU Script, invalid field names

END-START  
  *See* RMU Script, invalid field names

END-STRING  
  *See* RMU Script, invalid field names

END-SUBTRACT  
  *See* RMU Script, invalid field names

END-TRANSCIBE  
  *See* RMU Script, invalid field names

END-UNSTRING  
  *See* RMU Script, invalid field names

END-WRITE  
  *See* RMU Script, invalid field names

ENDG  
  *See* RMU Script, invalid field names

ENDING  
  *See* RMU Script, invalid field names

ENTER  
  *See* RMU Script, invalid field names

ENTRY  
  *See* RMU Script, invalid field names

ENUM1, formatting column values 21

ENVIRONMENT  
  *See* RMU Script, invalid field names

environmental errors 31

environments, UNIX 4

EOP  
  *See* RMU Script, invalid field names

EQ  
  *See* RMU Script, invalid field names

EQUAL  
  *See* RMU Script, invalid field names

EQUALS  
  *See* RMU Script, invalid field names

ERASE  
  *See* RMU Script, invalid field names

ERROR  
  *See* RMU Script, invalid field names

ESCAPE  
  *See* RMU Script, invalid field names

ESI  
  *See* RMU Script, invalid field names

ETEXT, formatting column values 21

EVALUATE  
  *See also* RMU Script, invalid field names  
  field name 54  
  statement 24

EVALUATE statement 26, 45

EVERY  
  *See* RMU Script, invalid field names

EXACT  
  *See* RMU Script, invalid field names

example, RMU Script 19

EXCEEDS  
  *See* RMU Script, invalid field names

EXCEPTION  
  *See* RMU Script, invalid field names

EXCESS-3  
  *See* RMU Script, invalid field names

EXCLUSIVE  
  *See* RMU Script, invalid field names

EXEC  
  *See* RMU Script, invalid field names

EXECUTE  
See RMU Script, invalid field names  
EXHIBIT  
See RMU Script, invalid field names  
EXIT  
See RMU Script, invalid field names  
EXTEND  
See RMU Script, invalid field names  
EXTERNAL  
See RMU Script, invalid field names  
EXTERNALLY-DESCRIBED-KEY  
See RMU Script, invalid field names

## F

FALSE  
See RMU Script, invalid field names  
FD  
See RMU Script, invalid field names  
features, RMU 1  
FETCH  
See RMU Script, invalid field names  
field headings, removing 21  
FILE  
See RMU Script, invalid field names  
file I/O errors 31  
FILE-CONTROL  
See RMU Script, invalid field names  
FILE-ID  
See RMU Script, invalid field names  
files  
COBLIST 5, 6  
CSV 5  
descriptions 7  
FJCONFIG 6  
FJDMAP0 6  
FJIDOC0 5, 6  
FJIRPT0 5, 6  
FJORPT0 5, 6  
FJSVC99 6  
FJUNIX0 6  
gif 5, 7, 8, 15, 16, 19, 26, 29, 35, 37, 38  
LKEDMAP 6  
required  
compiling RMU Script 5  
publishing to z/OS UNIX 6  
running RMU in "link and go"  
mode 6  
running RMU with compiled RMU  
Script 6  
running RMU without RMU Script  
program 5  
RMUERR1 6  
RMULIST 6  
STDERR 6  
STDOUT 6  
SYSIN 6  
SYSLMOD 6  
SYSOUT 5, 6  
UNIX 4  
FILLER  
See RMU Script, invalid field names  
FINAL  
See RMU Script, invalid field names  
FIND  
See RMU Script, invalid field names

FINISH  
See RMU Script, invalid field names  
FIRST  
See RMU Script, invalid field names  
FIXED  
See RMU Script, invalid field names  
FJCONFIG  
file description 7  
publishing to z/OS UNIX 6  
FJCONFIG DD 15  
FJCONFIG, pointing to httpd.conf file 5  
FJDMAP0 6  
FJDMAP0, file description 7  
FJIDOC0  
file description 7  
library 26  
link and go mode 4, 6  
running RMU 3  
running with compiled RMU  
Script 3, 6  
running without RMU Script 5  
FJIRPT0  
file description 7  
link and go mode 6  
running with compiled RMU  
Script 6  
running without RMU Script 5  
FJORPT0  
file description 7  
link and go mode 6  
running with compiled RMU  
Script 6  
running without RMU Script 5  
FJSVC99  
file description 7  
link and go mode 6  
required when compiling 6  
running with compiled RMU  
Script 6  
FJSYSR, DD name 14  
FJUNIX0  
file description 8  
publishing to z/OS UNIX 6  
sending output 4, 14  
FJUNIX1, file description 8  
float characters, removing 21  
font size, changing 30  
FOOTING  
See RMU Script, invalid field names  
FOR  
See RMU Script, invalid field names  
BACKGROUND-COLOR  
See RMU Script, invalid field names  
BACKGROUND-COLOUR  
See RMU Script, invalid field names  
FORM  
See RMU Script, invalid field names  
FORMAT  
See RMU Script, invalid field names  
formats  
HTML 4  
HTML1 4  
formatting column values 21  
fragments, syntax diagrams xi  
FREE  
See RMU Script, invalid field names

FROM  
See RMU Script, invalid field names  
FULL  
See RMU Script, invalid field names  
FUNCTION  
See RMU Script, invalid field names  
FZH#PROC  
locating 14  
tailoring 13  
FZH#PROC, &SYS1.SFZHZJCLS  
library 15  
FZH#READ, &SYS1.SFZHZJCLS  
library 15  
FZHASMOJ 12  
FZHASMPJ 14  
FZHBPARS 4, 8  
FZHDELICJ 11, 18  
FZHEBCSS 30  
FZHLINKJ 3, 5, 11  
FZHOPTAB 11, 12  
FZHPROCS 13, 14  
FZHRMU00 3  
FZHRMU00 program 3, 4, 13  
FZHRMU01 3  
FZHRMU01 program 5, 13  
FZHRMUJ0 3, 5, 11, 24  
FZHRMUJ1 3, 5, 11  
FZHRMUJ2 4, 5, 11  
FZHRMUS1 13  
FZHRMUX0 3, 5, 11, 15  
FZHRMUX1 3, 5, 11, 15  
FZHRMUX2 4, 5, 11, 15  
FZHUNIX1 4, 15

## G

GENERATE  
See RMU Script, invalid field names  
GET  
See RMU Script, invalid field names  
gif files 5, 7, 8, 15, 16, 19, 26, 29, 35, 37, 38  
GIVING  
See RMU Script, invalid field names  
GLOBAL  
See RMU Script, invalid field names  
GO  
See RMU Script, invalid field names  
GOBACK  
See RMU Script, invalid field names  
GREATER  
See RMU Script, invalid field names  
GROUP  
See RMU Script, invalid field names

## H

HDR  
See RMU Script, invalid field names  
HEADER  
See RMU Script, invalid field names  
HIGH-VALUE  
See also RMU Script, invalid field  
names  
field name 54  
parameter, DEFINE statement 34

HIGH-VALUES  
 See also RMU Script, invalid field names  
 field name 54  
 parameter, DEFINE statement 34  
 HIGHLIGHT  
 See RMU Script, invalid field names  
 how RMU works 1  
 HTML  
 document  
 creating 24  
 decorating 29  
 format 4  
 HTML0, parameter, PARM statement 33  
 HTML1  
 document, creating 26  
 format 4  
 option 8  
 parameter, PARM statement 33  
 httpd.conf 5

## I

I-O  
 See RMU Script, invalid field names  
 I-O-CONTROL  
 See RMU Script, invalid field names  
 I, system-defined field, description 51  
 I/O, RMU Script 17  
 I2, system-defined field, description 51  
 ID  
 See RMU Script, invalid field names  
 IDENTIFICATION  
 See RMU Script, invalid field names  
 IDW01  
 See RMU Script, invalid field names  
 IDW02  
 See RMU Script, invalid field names  
 IDW03  
 See RMU Script, invalid field names  
 IDW04  
 See RMU Script, invalid field names  
 IDW05  
 See RMU Script, invalid field names  
 IDW06  
 See RMU Script, invalid field names  
 IDW07  
 See RMU Script, invalid field names  
 IDW08  
 See RMU Script, invalid field names  
 IDX  
 subscript  
 BYPASS statement 41  
 control breaks selection tree 26  
 creating CSV documents 21  
 creating simple HTML documents 24  
 description 51  
 EVALUATE statement 46  
 I/O handling 18  
 system-defined field, description 51  
 IDX2, system-defined field, description 51  
 IF  
 See also RMU Script, invalid field names  
 field name 55

IF (continued)  
 statement 47  
 IN  
 See RMU Script, invalid field names  
 INDEX  
 See RMU Script, invalid field names  
 INDEX-1  
 See RMU Script, invalid field names  
 INDEX-2  
 See RMU Script, invalid field names  
 INDEX-3  
 See RMU Script, invalid field names  
 INDEX-4  
 See RMU Script, invalid field names  
 INDEX-5  
 See RMU Script, invalid field names  
 INDEX-6  
 See RMU Script, invalid field names  
 INDEX-7  
 See RMU Script, invalid field names  
 INDEX-8  
 See RMU Script, invalid field names  
 INDEX-9  
 See RMU Script, invalid field names  
 INDEXED  
 See RMU Script, invalid field names  
 INDIC  
 See RMU Script, invalid field names  
 INDICATE  
 See RMU Script, invalid field names  
 INDICATOR  
 See RMU Script, invalid field names  
 INDICATORS  
 See RMU Script, invalid field names  
 INITIAL  
 See RMU Script, invalid field names  
 INITIALIZE  
 See RMU Script, invalid field names  
 INITIATE  
 See RMU Script, invalid field names  
 INPUT  
 See RMU Script, invalid field names  
 INPUT-OUTPUT  
 See RMU Script, invalid field names  
 INSERT  
 See RMU Script, invalid field names  
 INSPECT  
 See RMU Script, invalid field names  
 INSTALLATION  
 See RMU Script, invalid field names  
 installation, RMU 11  
 INTO  
 See RMU Script, invalid field names  
 introduction, RMU 1  
 INVALID  
 See RMU Script, invalid field names  
 IS  
 See RMU Script, invalid field names  
 IW-FAKE-INDEX  
 See RMU Script, invalid field names  
 IW-PENGI-PAGE  
 See RMU Script, invalid field names  
 IW-RETURN-CODE  
 See RMU Script, invalid field names  
 IW-WORK-INDEX  
 See RMU Script, invalid field names

## J

JAPANESE  
 See RMU Script, invalid field names  
 Java 4, 8, 9, 16  
 JCL, supplied with RMU 3  
 jobs  
 available 5  
 FZHASMOJ 12  
 FZHASMPJ 14  
 FZHLINKJ 3  
 FZHRMUJ0 3, 5  
 FZHRMUJ1 3, 5  
 FZHRMUJ2 4, 5  
 FZHRMUX0 3, 15  
 FZHRMUX1 3, 15  
 FZHRMUX2 4, 15  
 JUST  
 See RMU Script, invalid field names  
 JUSTIFIED  
 See RMU Script, invalid field names

## K

KANJI  
 See RMU Script, invalid field names  
 KEEP  
 See RMU Script, invalid field names  
 KEPT  
 See RMU Script, invalid field names  
 KEY  
 See RMU Script, invalid field names  
 KEYBOARD  
 See RMU Script, invalid field names  
 keywords, syntax diagrams xi

## L

LAST  
 See RMU Script, invalid field names  
 LABEL  
 See RMU Script, invalid field names  
 LD  
 See RMU Script, invalid field names  
 LEADING  
 See RMU Script, invalid field names  
 LEFT  
 See RMU Script, invalid field names  
 LEFT-JUSTIFY  
 See RMU Script, invalid field names  
 LENGTH  
 See RMU Script, invalid field names  
 LENGTH-CHECK  
 See RMU Script, invalid field names  
 LESS  
 See RMU Script, invalid field names  
 libraries  
 &SYS1.SFZHAMAC 16  
 &SYS1.SFZHASRC 11, 14, 16  
 &SYS1.SFZHDOCS 7, 9, 16, 30, 35  
 &SYS1.SFZHJCLS 3, 4, 5, 11, 12, 13, 14, 15, 18, 19, 21, 24, 26, 29  
 &SYS1.SFZHLOAD 14  
 &SYS1.SFZHPROC 15  
 FJIDOC0 26  
 library  
 jobs 15



library (*continued*)  
 locator table, default 13  
 LIKE  
 See RMU Script, invalid field names  
 LIMIT  
 See RMU Script, invalid field names  
 LIMITS  
 See RMU Script, invalid field names  
 LINAGE  
 See RMU Script, invalid field names  
 LINAGE-COUNTER  
 See RMU Script, invalid field names  
 LINE  
 parameter, BYPASS statement 41  
 system-defined field, description 51  
 line number 18  
 LINE-COUNT  
 determining page properties 21  
 EVALUATE statement 46  
 parameter 24, 26  
 system-defined field, description 51  
 LINE-COUNTER  
 See RMU Script, invalid field names  
 LINE1 thru LINE66, system-defined field,  
 description 51  
 LINES  
 See RMU Script, invalid field names  
 linex, parameter, EVALUATE  
 statement 46  
 link and go  
 FZHRMU00 16  
 FZHRMU00 program 13  
 FZHRMU00 utility 15  
 FZHRMUJ2 5  
 FZHRMUJ2 job 11  
 FZHRMUJ2, FZHRMUX2 jobs 4  
 FZHRMUX2 job 5, 11  
 FZHTEST0 15  
 FZHTEST1 15  
 FZHTEST2 15  
 FZHTEST3 15  
 FZHTEST4 15  
 required files 6  
 RMU default library locator table  
 (FZHPROCS) 13  
 RMU Script, basic concepts 17  
 running RMU 4  
 LINKAGE  
 See RMU Script, invalid field names  
 LIST option 12  
 LIT  
 See RMU Script, invalid field names  
 LKED option 8  
 LKEDMAP  
 file 6  
 file description 8  
 required when compiling 6  
 LOCALLY  
 See RMU Script, invalid field names  
 locating statement in error 31  
 LOCK  
 See RMU Script, invalid field names  
 LOW-VALUE  
 See also RMU Script, invalid field  
 names  
 field name 55  
 parameter, DEFINE statement 34

LOW-VALUES  
 See also RMU Script, invalid field  
 names  
 field name 55  
 parameter, DEFINE statement 34

## M

MANUAL  
 See RMU Script, invalid field names  
 manual, structure ix  
 MASK  
 See RMU Script, invalid field names  
 MEMBER  
 See RMU Script, invalid field names  
 members  
 FZH#PROC 14  
 FZHEBCSS 30  
 MEMORY  
 See RMU Script, invalid field names  
 MERGE  
 See RMU Script, invalid field names  
 MESSAGE  
 See RMU Script, invalid field names  
 messages 59  
 Migration Utility  
 DD name, FJSYSR 14  
 libraries 14  
 MODE  
 See RMU Script, invalid field names  
 MODEL parameter 14  
 MODIFIED  
 See RMU Script, invalid field names  
 MODIFY  
 See RMU Script, invalid field names  
 MODULES  
 See RMU Script, invalid field names  
 MORE-LABELS  
 See RMU Script, invalid field names  
 MOVE  
 See RMU Script, invalid field names  
 MSGALL option 14  
 MSGOFF option 14  
 MSGON option 14  
 MSGSER option 14  
 MSGTXT option 14  
 MULTIPLE  
 See RMU Script, invalid field names  
 MULTIPLY  
 See RMU Script, invalid field names

## N

N, parameter, DEFINE statement 34  
 NATIVE  
 See RMU Script, invalid field names  
 NEGATIVE  
 See RMU Script, invalid field names  
 NEXT  
 See RMU Script, invalid field names  
 NIX  
 directory 5  
 environment 4  
 files 4

NO  
 See also RMU Script, invalid field  
 names  
 parameter, PARM statement 33  
 TURF option 3  
 NO-ECHO  
 See RMU Script, invalid field names  
 NONAME, default value, PARM  
 statement 33  
 NONE  
 See RMU Script, invalid field names  
 NORMAL  
 See RMU Script, invalid field names  
 NOT  
 See RMU Script, invalid field names  
 Notices 65  
 NULL  
 See RMU Script, invalid field names  
 NULLS  
 See RMU Script, invalid field names  
 NUMBER  
 See RMU Script, invalid field names  
 NUMERIC  
 See RMU Script, invalid field names  
 NUMERIC-EDITED  
 See RMU Script, invalid field names

## O

OBJECT  
 See RMU Script, invalid field names  
 object statement 35  
 OBJECT-COMPUTER  
 See RMU Script, invalid field names  
 OCCURS  
 See RMU Script, invalid field names  
 OF  
 See RMU Script, invalid field names  
 OFF  
 See RMU Script, invalid field names  
 OMITTED  
 See RMU Script, invalid field names  
 ON  
 See RMU Script, invalid field names  
 ONLY  
 See RMU Script, invalid field names  
 OPEN  
 See RMU Script, invalid field names  
 OPTIONAL  
 See RMU Script, invalid field names  
 options  
 1403-paper 3, 33, 36  
 COBOL 7  
 COBOL compiler PROCESS (CBL) 49  
 HTML1 8  
 LIST 12  
 LKED 8  
 MSGALL 14  
 MSGOFF 14  
 MSGON 14  
 MSGSER 14  
 MSGTXT 14  
 NO (TURF option) 3  
 STYLE 3  
 SYSPRDD=&DDname 14  
 table, RMU 11  
 TURF 3

OR  
*See* RMU Script, invalid field names  
ORDER  
*See* RMU Script, invalid field names  
ORGANIZATION  
*See* RMU Script, invalid field names  
OTHER  
*See also* RMU Script, invalid field names  
field name 55  
parameter, EVALUATE statement 46  
OUTPUT  
*See* RMU Script, invalid field names  
output documents, distributing 4  
OVERFLOW  
*See* RMU Script, invalid field names  
OWNER  
*See* RMU Script, invalid field names

## P

P, parameter, DEFINE statement 34  
PACKED-DECIMAL  
*See* RMU Script, invalid field names  
PADDING  
*See* RMU Script, invalid field names  
page  
properties, determining 21  
titles, removing 21  
PAGE  
*See* RMU Script, invalid field names  
page\_end statement 38  
page\_top statement 37  
PAGE-COUNT  
determining page properties 21  
parameter 24, 26  
system-defined field, description 51  
PAGE-COUNTER  
*See* RMU Script, invalid field names  
PAGEFOOT  
*See* RMU Script, invalid field names  
PALETTE  
*See* RMU Script, invalid field names  
PARAGRAPH  
*See* RMU Script, invalid field names  
parameters  
.ENUM1 40  
.ETEXT 40  
&char 33, 50  
&color 36  
&condition 45, 46  
&count 45  
&css 3, 4  
&currency 33  
&decimal 33  
&expression 47  
&field 34, 42, 43, 44, 46, 50  
&format 33  
&formula 39  
&HTML0 33  
&image 35, 41  
&length 34, 39, 40, 44, 46  
&line 40  
&LINEn 41  
&literal 50  
&mod 35  
&n 36

parameters (*continued*)  
&name 33  
&option 49  
&pointer 50  
&procname 48  
&program 42  
&recfield 39  
&sendfield 39  
&size 36  
&start 39, 40, 44, 46  
&style 40  
&target 50  
&type 34  
&value 34  
1403-paper 33, 36  
A 34  
B 34  
COBLIB 14  
COMMA 33  
CSVCHAR 13  
CURRENCY 11  
CWORK 14  
DECIMAL 11  
HIGH-VALUE 34  
HIGH-VALUES 34  
HTML1 33  
LINE 41  
LINE-COUNT 24, 26  
linex 46  
LOW-VALUE 34  
LOW-VALUES 34  
MODEL 14  
N 34  
NO 33  
OTHER 46  
P 34  
PAGE-COUNT 24, 26  
PATH 14  
PERIOD 33  
SPACE 34  
SPACES 34  
statements\_1 47  
statements\_2 47  
SVC99 14  
TURF NO 30  
TWORK 14  
UNTIL 45  
WHILE 45  
ZERO 34  
ZEROES 34  
PARM statement 24, 26, 33  
PARM-LENGTH  
*See* RMU Script, invalid field names  
PARM-REGISTER  
*See* RMU Script, invalid field names  
PARM-REGISTER-9  
*See* RMU Script, invalid field names  
PASSWORD  
*See* RMU Script, invalid field names  
PATH parameter 14  
PCs, downloading to 5  
PDS  
FJIDOC0 3, 4  
PENGI-CODE0  
*See* RMU Script, invalid field names  
PENGI-CODES  
*See* RMU Script, invalid field names

PENGI-STATUS  
*See* RMU Script, invalid field names  
PERFORM  
*See also* RMU Script, invalid field names  
field name 56  
statement 47  
PERIOD, parameter, PARM statement 33  
PF  
*See* RMU Script, invalid field names  
PH  
*See* RMU Script, invalid field names  
PIC  
*See* RMU Script, invalid field names  
PICTURE  
*See* RMU Script, invalid field names  
PLUS  
*See* RMU Script, invalid field names  
POINTER  
*See* RMU Script, invalid field names  
POS  
*See* RMU Script, invalid field names  
POSITION  
*See* RMU Script, invalid field names  
POSITIVE  
*See* RMU Script, invalid field names  
PRESENT  
*See* RMU Script, invalid field names  
PREVIOUS  
*See* RMU Script, invalid field names  
print control character 18  
PRINT-SWITCH  
*See* RMU Script, invalid field names  
PRINTER  
*See* RMU Script, invalid field names  
PRINTER-1  
*See* RMU Script, invalid field names  
printer, internal 14  
PRINTING  
*See* RMU Script, invalid field names  
PRIOR  
*See* RMU Script, invalid field names  
problems, runtime 31  
PROC statement 48  
PROCEDURE  
*See* RMU Script, invalid field names  
PROCEDURES  
*See* RMU Script, invalid field names  
PROCEED  
*See* RMU Script, invalid field names  
PROCESS  
*See* RMU Script, invalid field names  
PROCESS statement 49  
PROCESSING  
*See* RMU Script, invalid field names  
PROCLIB0 constant 14  
PRODUCT0 constant 14  
PROGRAM  
*See* RMU Script, invalid field names  
program check interrupts 31  
Program Directory 11  
PROGRAM-DATE  
*See* RMU Script, invalid field names  
PROGRAM-ID  
*See* RMU Script, invalid field names  
PROGRAM-INFO-TABLE  
*See* RMU Script, invalid field names

PROGRAM-NAME  
*See* RMU Script, invalid field names  
PROGRAM-OPYSYS  
*See* RMU Script, invalid field names  
PROGRAM-TIME  
*See* RMU Script, invalid field names  
PROGRAM-TYPE  
*See* RMU Script, invalid field names  
programs  
FZHBPARS 8  
FZHLINKJ 5  
FZHPROCS 14  
FZHRMU00 3, 4, 13  
FZHRMU01 3, 5, 13  
FZHUNIX1 4, 15  
PROMPT  
*See* RMU Script, invalid field names  
PROTECTED  
*See* RMU Script, invalid field names  
PURGE  
*See* RMU Script, invalid field names

## Q

QUEUE  
*See* RMU Script, invalid field names  
QUOTE  
*See* RMU Script, invalid field names  
QUOTES  
*See* RMU Script, invalid field names

## R

RANDOM  
*See* RMU Script, invalid field names  
RANGE  
*See* RMU Script, invalid field names  
RC  
*See* RMU Script, invalid field names  
RC-ABEND00  
*See* RMU Script, invalid field names  
RC-ABRCODE  
*See* RMU Script, invalid field names  
RC-BOOLEAN  
*See* RMU Script, invalid field names  
RC-CHKOVF0  
*See* RMU Script, invalid field names  
RC-COBSTAT  
*See* RMU Script, invalid field names  
RC-DATESWP  
*See* RMU Script, invalid field names  
RC-HEXSTRO  
*See* RMU Script, invalid field names  
RC-HEXSTR1  
*See* RMU Script, invalid field names  
RC-REPCHR0  
*See* RMU Script, invalid field names  
RD  
*See* RMU Script, invalid field names  
READ  
*See* RMU Script, invalid field names  
READY  
*See* RMU Script, invalid field names  
REALM  
*See* RMU Script, invalid field names

RECEIVE  
*See* RMU Script, invalid field names  
RECONNECT  
*See* RMU Script, invalid field names  
RECORD  
*See* RMU Script, invalid field names  
RECORD-NAME  
*See* RMU Script, invalid field names  
RECORDING  
*See* RMU Script, invalid field names  
RECORDS  
*See* RMU Script, invalid field names  
REDEFINES  
*See* RMU Script, invalid field names  
REEL  
*See* RMU Script, invalid field names  
REFERENCE  
*See* RMU Script, invalid field names  
REFERENCES  
*See* RMU Script, invalid field names  
RELATION  
*See* RMU Script, invalid field names  
RELATIVE  
*See* RMU Script, invalid field names  
RELEASE  
*See* RMU Script, invalid field names  
RELOAD  
*See* RMU Script, invalid field names  
REMAINDER  
*See* RMU Script, invalid field names  
REMOVAL  
*See* RMU Script, invalid field names  
removing lines 21  
RENAMES  
*See* RMU Script, invalid field names  
repeatable items, syntax diagrams xi  
REPEATED  
*See* RMU Script, invalid field names  
REPLACE  
*See* RMU Script, invalid field names  
REPLACING  
*See* RMU Script, invalid field names  
REPORT  
*See* RMU Script, invalid field names  
REPORTING  
*See* RMU Script, invalid field names  
REPORTS  
*See* RMU Script, invalid field names  
REQUIRED  
*See* RMU Script, invalid field names  
RERUN  
*See* RMU Script, invalid field names  
RESERVE  
*See* RMU Script, invalid field names  
reserved  
fields, COBOL 52  
RESET  
*See* RMU Script, invalid field names  
RETAINING  
*See* RMU Script, invalid field names  
RETRIEVAL  
*See* RMU Script, invalid field names  
RETURN  
*See* RMU Script, invalid field names  
RETURN-CODE  
STOP statement 49  
system-defined field, description 51

REVERSE-VIDEO  
*See* RMU Script, invalid field names  
REVERSED  
*See* RMU Script, invalid field names  
REWIND  
*See* RMU Script, invalid field names  
REWRITE  
*See* RMU Script, invalid field names  
RF  
*See* RMU Script, invalid field names  
RH  
*See* RMU Script, invalid field names  
RIGHT  
*See* RMU Script, invalid field names  
RIGHT-JUSTIFY  
*See* RMU Script, invalid field names  
RMU  
compiler generated messages 59  
default  
library locator table 13  
options table 11  
features 1  
how it works 1  
installation 11  
introduction 1  
libraries 11  
messages  
compiler generated 59  
runtime, error 59  
reserved words 16  
running  
in link and go mode 6  
JCL, supplied with RMU 3  
with compiled RMU Script 6  
without RMU Script program 5  
runtime error messages 59  
sections, description 17  
ways of running 3  
working with 3  
RMU Script  
available jobs 5  
coding rules 18  
compile and link 5  
compiler jobs 5  
compiling, required files 5  
concepts 17  
CSV document 21  
debugging 30  
example 19  
HTML document  
creating simple 24  
creating with control breaks  
selection tree 26  
HTML document, decorating 29  
I/O 17  
invalid field names 52  
language instruction reference 33  
working with 17  
RMUERR1  
file description 8  
link and go 6  
required when compiling 6  
RMULIST  
description 8  
link and go 6  
required when compiling 6

ROLLBACK  
*See* RMU Script, invalid field names  
 ROLLING  
*See* RMU Script, invalid field names  
 ROUNDED  
*See* RMU Script, invalid field names  
 RUN  
*See* RMU Script, invalid field names  
 runtime problems 31

## S

SAME  
*See* RMU Script, invalid field names  
 SCREEN  
*See* RMU Script, invalid field names  
 SD  
*See* RMU Script, invalid field names  
 SEARCH  
*See* RMU Script, invalid field names  
 SECTION  
*See* RMU Script, invalid field names  
 sections, RMU 17  
 SECURE  
*See* RMU Script, invalid field names  
 SECURITY  
*See* RMU Script, invalid field names  
 SEGMENT  
*See* RMU Script, invalid field names  
 SEGMENT-LIMIT  
*See* RMU Script, invalid field names  
 SEL  
*See* RMU Script, invalid field names  
 SELECT  
*See* RMU Script, invalid field names  
 SEND  
*See* RMU Script, invalid field names  
 SENTENCE  
*See* RMU Script, invalid field names  
 SEPARATE  
*See* RMU Script, invalid field names  
 SEQUENCE  
*See* RMU Script, invalid field names  
 sequence numbers, types 30  
 SEQUENTIAL  
*See* RMU Script, invalid field names  
 SERVICE  
*See* RMU Script, invalid field names  
 SESSION-ID  
*See* RMU Script, invalid field names  
 SET  
*See* RMU Script, invalid field names  
 SHARED  
*See* RMU Script, invalid field names  
 SHIFT-IN  
*See* RMU Script, invalid field names  
 SHIFT-OUT  
*See* RMU Script, invalid field names  
 SIGN  
*See* RMU Script, invalid field names  
 SIZE  
*See* RMU Script, invalid field names  
 SKIP1  
*See* RMU Script, invalid field names  
 SKIP2  
*See* RMU Script, invalid field names

SKIP3  
*See* RMU Script, invalid field names  
 SMP/E 11  
 SORT  
*See* RMU Script, invalid field names  
 SORT-CONTROL  
*See* RMU Script, invalid field names  
 SORT-CORE-SIZE  
*See* RMU Script, invalid field names  
 SORT-FILE-SIZE  
*See* RMU Script, invalid field names  
 SORT-MERGE  
*See* RMU Script, invalid field names  
 SORT-MESSAGE  
*See* RMU Script, invalid field names  
 SORT-MODE-SIZE  
*See* RMU Script, invalid field names  
 SORT-RETURN  
*See* RMU Script, invalid field names  
 SOURCE  
*See* RMU Script, invalid field names  
 SOURCE-COMPUTER  
*See* RMU Script, invalid field names  
 SPACE  
*See also* RMU Script, invalid field names  
 field name 57  
 parameter, DEFINE statement 34  
 SPACE-FILL  
*See* RMU Script, invalid field names  
 SPACES  
*See also* RMU Script, invalid field names  
 field name 57  
 parameter, DEFINE statement 34  
 SPECIAL-NAMES  
*See* RMU Script, invalid field names  
 STANDARD  
*See* RMU Script, invalid field names  
 STANDARD-1  
*See* RMU Script, invalid field names  
 STANDARD-2  
*See* RMU Script, invalid field names  
 STANDARD-3  
*See* RMU Script, invalid field names  
 STANDARD-4  
*See* RMU Script, invalid field names  
 START  
*See* RMU Script, invalid field names  
 STARTING  
*See* RMU Script, invalid field names  
 statement sequence numbers 30  
 statements  
 <docs> 36  
 <object> 35  
 <page\_end> 38  
 <page\_top> 37  
 <style> 35  
 assignment 38  
 BYPASS 21, 24, 26, 41  
 CALL 42  
 COBOL 52  
 CONTINUE 42  
 CONTROL 26, 43  
 DEFINE 26, 34  
 DISPLAY 44  
 DO 44

statements (*continued*)  
 docs\_end 37  
 ELSE 47  
 END-DO 44  
 END-EVALUATE 45  
 END-IF 47  
 END-PROC 48  
 EVALUATE 24, 26, 45  
 IF 47  
 PARM 24, 26, 33  
 PERFORM 47  
 PROC 48  
 PROCESS 49  
 STOP 49  
 STRING 50  
 statements\_1, parameter, IF statement 47  
 statements\_2, parameter, IF statement 47  
 STATUS  
*See* RMU Script, invalid field names  
 STDERR  
 file description 8  
 publishing to z/OS UNIX 6  
 STDOUT  
 file description 8  
 publishing to z/OS UNIX 6  
 STOP  
*See* RMU Script, invalid field names  
 STOP statement 49  
 STORE  
*See* RMU Script, invalid field names  
 STRING  
*See also* RMU Script, invalid field names  
 field name 57  
 statement 50  
 STYLE  
*See also* RMU Script, invalid field names  
 field name 57  
 option 3  
 style statement 35  
 SUB-QUEUE-1  
*See* RMU Script, invalid field names  
 SUB-QUEUE-2  
*See* RMU Script, invalid field names  
 SUB-QUEUE-3  
*See* RMU Script, invalid field names  
 SUB-SCHEMA  
*See* RMU Script, invalid field names  
 SUBFILE  
*See* RMU Script, invalid field names  
 SUBPROGRAM  
*See* RMU Script, invalid field names  
 SUBTRACT  
*See* RMU Script, invalid field names  
 SUM  
*See* RMU Script, invalid field names  
 SUPPRESS  
*See* RMU Script, invalid field names  
 SVC99 parameter 14  
 SWITCH  
*See* RMU Script, invalid field names  
 SWITCH-1  
*See* RMU Script, invalid field names  
 SWITCH-2  
*See* RMU Script, invalid field names

SWITCH-3  
*See* RMU Script, invalid field names

SWITCH-4  
*See* RMU Script, invalid field names

SWITCH-5  
*See* RMU Script, invalid field names

SWITCH-6  
*See* RMU Script, invalid field names

SWITCH-7  
*See* RMU Script, invalid field names

SWITCH-8  
*See* RMU Script, invalid field names

SYMBOL, &sys1 14

SYMBOLIC  
*See* RMU Script, invalid field names

SYNC  
*See* RMU Script, invalid field names

SYNCHRONIZED  
*See* RMU Script, invalid field names

syntax diagrams, how to read xi

SYSDATE or SYSDATE-E, system-defined field, description 51

SYSDATE-9, system-defined field, description 51

SYSDATE-LONG or SYSDATE-LONG-E, system-defined field, description 52

SYSDATE-LONG-9, system-defined field, description 52

SYSIN  
 compiling RMU Script 6  
 file description 8  
 link and go 6  
 link and go mode 4  
 not coding in JCL 3

SYSLMOD, required when compiling 6

SYSOUT  
 DISPLAY statement 44  
 file description 8  
 link and go 6  
 running with compiled RMU Script 6  
 running without RMU Script 5

SYSRDD=&DDname option 14

system  
 defined fields 51  
 information 15

SYSTIME-9, system-defined field, description 52

SYSTIME, system-defined field, description 52

## T

TABLE  
*See* RMU Script, invalid field names

table, FZHPROCS 14

TALLY  
*See* RMU Script, invalid field names

TALLYING  
*See* RMU Script, invalid field names

TAPE  
*See* RMU Script, invalid field names

TENANT  
*See* RMU Script, invalid field names

TERMINAL  
*See* RMU Script, invalid field names

TERMINATE  
*See* RMU Script, invalid field names

TEST  
*See* RMU Script, invalid field names

TEXT  
*See* RMU Script, invalid field names

text, highlighting 30

THAN  
*See* RMU Script, invalid field names

THEN  
*See* RMU Script, invalid field names

THROUGH  
*See* RMU Script, invalid field names

THRU  
*See* RMU Script, invalid field names

TIME  
*See* RMU Script, invalid field names

TIMEOUT  
*See* RMU Script, invalid field names

TIMES  
*See* RMU Script, invalid field names

TITLE  
*See* RMU Script, invalid field names

TO  
*See* RMU Script, invalid field names

TOP  
*See* RMU Script, invalid field names

TRACE  
*See* RMU Script, invalid field names

trademarks 66

TRAILING  
*See* RMU Script, invalid field names

TRAILING-SIGN  
*See* RMU Script, invalid field names

TRANSACTION  
*See* RMU Script, invalid field names

TRANSCEIVE  
*See* RMU Script, invalid field names

TRUE  
*See* RMU Script, invalid field names

TURF option 3, 13, 30, 36

TWORK parameter 14

TYPE  
*See* RMU Script, invalid field names

## U

UNDERLINE  
*See* RMU Script, invalid field names

UNEQUAL  
*See* RMU Script, invalid field names

UNIT  
*See* RMU Script, invalid field names

UNIX, setting up environment 14

UNLOCK  
*See* RMU Script, invalid field names

UNSTRING  
*See* RMU Script, invalid field names

UNTIL  
*See also* RMU Script, invalid field names

field name 57

parameter, DO statement 45

UP  
*See* RMU Script, invalid field names

UPDATE  
*See* RMU Script, invalid field names

UPON  
*See* RMU Script, invalid field names

USAGE  
*See* RMU Script, invalid field names

USAGE-MODE  
*See* RMU Script, invalid field names

USE  
*See* RMU Script, invalid field names

USER  
*See* RMU Script, invalid field names

USING  
*See* RMU Script, invalid field names

## V

VAL  
*See* RMU Script, invalid field names

VALID  
*See* RMU Script, invalid field names

VALIDATE  
*See* RMU Script, invalid field names

VALUE  
*See* RMU Script, invalid field names

VALUES  
*See* RMU Script, invalid field names

VARCHAR  
*See* RMU Script, invalid field names

VARIABLE  
*See* RMU Script, invalid field names

variables, syntax diagrams xi

VARYING  
*See* RMU Script, invalid field names

verbs, COBOL 52

## W

WAIT  
*See* RMU Script, invalid field names

WHEN  
*See* RMU Script, invalid field names

WHEN-COMPILED  
*See* RMU Script, invalid field names

WHILE, parameter, DO statement 45

WITH  
*See* RMU Script, invalid field names

WITHIN  
*See* RMU Script, invalid field names

WORDS  
*See* RMU Script, invalid field names

working with RMU 3

WORKING-STORAGE  
*See* RMU Script, invalid field names

WRITE  
*See* RMU Script, invalid field names

WRITE-ONLY  
*See* RMU Script, invalid field names

## Z

z/OS Enterprise COBOL 13

z/OS server  
 publishing to 14  
 setting up environment 14

ZERO  
*See also* RMU Script, invalid field names

ZERO (*continued*)  
parameter, DEFINE statement 34  
ZERO-FILL  
*See* RMU Script, invalid field names  
ZEROES  
*See also* RMU Script, invalid field  
names  
field name 58  
parameter, DEFINE statement 34  
ZEROS  
*See also* RMU Script, invalid field  
names  
field name 58

---

# Readers' Comments — We'd Like to Hear from You

Report Modernization Utility for z/OS and OS/390  
User's Guide and Reference  
Version 1 Release 1

Publication No. SC19-2726-00

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Send your comments to the address on the reverse side of this form.

If you would like a response from IBM, please fill in the following information:

\_\_\_\_\_

Name

\_\_\_\_\_

Address

\_\_\_\_\_

Company or Organization

\_\_\_\_\_

Phone No.

\_\_\_\_\_

E-mail address



Fold and Tape

Please do not staple

Fold and Tape



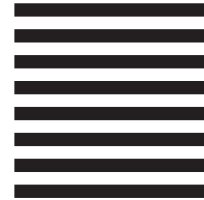
NO POSTAGE  
NECESSARY  
IF MAILED IN THE  
UNITED STATES

# BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation  
Reader Comments  
DTX/E269  
555 Bailey Avenue  
San Jose, CA  
U.S.A. 95141-9989



Fold and Tape

Please do not staple

Fold and Tape







Program Number: 5697-N44

Printed in USA

SC19-2726-00



Spine information:



Report Modernization Utility for  
z/OS and OS/390

Report Modernization Utility for z/OS and OS/390  
VIR1 User's Guide

Version 1  
Release 1