



*JES2MAIL with
Report Processing Language (RPL)
PDF Form Overlay*

JES²Mail

Quick Start

z/OS
OS/390

Documentation Release 4.3.1

April 1, 2005

Copyright and Ownership:

Every effort has been made to supply complete and accurate information. However, LaBayne and Associates Inc, and Computer Application Services, Inc. assume no responsibility for its use, nor for any infringement of patents or other rights of third parties which would result.

JES2MAIL is a proprietary product to be used only according to the terms and conditions of sale, lease or subscription. All software is the exclusive property of LaBayne and Associates, Fountain Valley, California.

Copyright 1999-2005, LaBayne and Associates, Inc. All Rights Reserved

No part of this publication may be stored in a retrieval system, transmitted, or reproduced in any way, including but not limited to photocopy, photograph, magnetic or other record, without the prior written agreement and written permission of the publisher.

Sales & Support:

Sales and support are provided by:

Computer Application Services Inc. (CASI)
10231 Slater Ave.
Fountain Valley, CA 90708

www.casisoft.com

(714) 378-1109 Sales (sales@casisoft.com)
(714) 378-0208 Support (support@casisoft.com)
(714) 378-9909 Fax

Contents

<i>Chapter 1</i>	Introduction	1
	Overall Description	1
	Scope of the Quick Start Manual	1
<i>Chapter 2</i>	Basic Installation	3
	Installation Task list:	3
	Product update.	3
	Step 1: PKUNZIP the Distribution 'PC' File.....	4
	Step 2: Transfer XMITDIST and RCVINST files to host.....	5
	The allocate space for the uploaded file	5
	The Upload XMITDIST and RCVINST files.	5
	Step 3: Use RCVINST job to unload the libraries	6
	Step 4: Linkedit JES2Mail	8
	Supplied object modules.....	8
	Linkedit JES2MAIL	8
	Step 5: Run the Installation Verification Job	9
	Step 6: Optionally Write Batch Mode Procedures.....	11
	Step 7: Create an Initial "Configuration File" Member	11
	Execution Prerequisites	12
	RACF Authorization	12
<i>Chapter 3</i>	System Configuration	13
	Configuration Statements	13
	/INSERT instruction	13
	AuthCode: (Required).....	14
	LocalCodePage:	14
	LocalZone:	15
	OurDomainName: (Required).....	15
	OverlayAuthCode: (Required for Form Overlay)	15
	PODown:	15
	PostOffice: (Required)	15
	SetEnv:	16
	SplitAuthCode: (Required for Report Splitter).....	16
	Trace:	16
	ZipArchiveSize:.....	17
	ZIPService:	17
	ZipWorkBufferSize:.....	17
<i>Chapter 4</i>	Batch Job Mode	19
	Overview.....	19
	JCL	19

	Command Input	20
	FILE command	20
	TRACE command	20
	REPORT command	20
	RULESET command	21
	FORMNAME command	21
	Examples:	21
	Send a report in Batch mode	21
	Send a report with inline RuleSet	22
Chapter 5	Simple Message File Mode	23
	Overview	23
	SMTP Message Header statements	23
	Message/File Header Statements	24
	Insert additional files	26
	Symbolic Substitution	26
	Escape Character	27
	Word Wrap	27
	Zipping Attachments	28
	Sample Jobs	30
	Send a Message File with inline addressing	30
	Send a Message File with attachments	30
	All in one JOB Example	31
	Example with word wrap	31
	Example of HTML and variable substitution	32
Chapter 6	Report Processing	33
	Overview	33
	The Default Rule	34
	Routing Statements	35
	Route Lists	35
	FileNames in configuration statements	35
	Delivered Report Formats (PackageType)	36
	ASA and MCC Carriage Control Handling	38
	Symbolic Substitution	38
	Report RuleSet Statements	39
	/INSERT instruction	39
	Action:	40
	AttachAsName:	40
	BCC:	40
	CC:	41
	CoverPage:	41
	DOCInfoAuthor:	41
	DOCInfoCreator:	41
	DOCInfoKeywords:	42
	DOCInfoProducer:	42
	DOCInfoSubject:	42
	DOCInfoTitle:	42
	Font:	42
	FontSize:	43
	From:	43
	ID: (Required)	44
	LinesPerInch:	44

MaxLinesPerPage:	44
Name:	44
Outline:	44
Overlay:	45
PackageType:	45
PageHeight:	46
PageSize:	46
PageWidth:	46
PDFCenterWindow:	46
ProtectLevel:	46
ProtectOp:	47
ProtectOwnerPassword:	47
ProtectUserPassword:	48
Sanitize:	48
SuppressCC:	49
To:	49
ZIPMemberName	49
E-mail Addressing	49
Route Lists	50

<i>Chapter 7</i>	ZIP Compression Support	51
	Configuration File	51
	Ruleset	52
	ATTACH statement	52
	Native Compression Support	54
	Examples:	55
	Example of JES2Mail ATTACHments	55
	Example of JES2Mail Main Report	57
	Readers' Comments – We'd Like to Hear from You	59

1

Introduction

Overall Description

The JES2MAIL product is designed to take reports from a z/OS or OS/390 host and send them to an SMTP compliant e-mail “post office”. Reports can be encoded and delivered in Adobe PDF (Portable Document Format), HTML (Hyper Text Markup Language), RTF (Rich Text Format) or ‘just plain text’ formats. The reports can come into JES2Mail from the JES queue, from an HFS directory, or via a file passed in a batch job.

With the **report split** option, RPL, JES2MAIL can “chop up” reports and e-mail the separate pieces. The report splitting process is controlled by a robust script language.

A second mode of the product is to process sequential datasets that contain preformatted messages. The messages can request that attachments be included in the e-mail message sent to the post office. The ‘Message File’ can contain any number of e-mail messages.

Messages can be addressed to specific destinations, or the destination address can refer to a distribution list file that can contain a large number of destinations. (Only limited by the capability of the post office.)

Scope of the Quick Start Manual

This manual covers the minimum steps to install and use the JES2Mail product. Besides the installation process a set of the most common configuration and report processing parameters are discussed. For more complete information refer to the full JES2Mail Installand and Reference manual.

2

Basic Installation

JES2MAIL is delivered as a PC file comprising both PC and OS/390 parts. You will upload the OS/390 parts, linkedit the system, tailor a configuration file, and optionally create other files that direct the processing of particular reports.

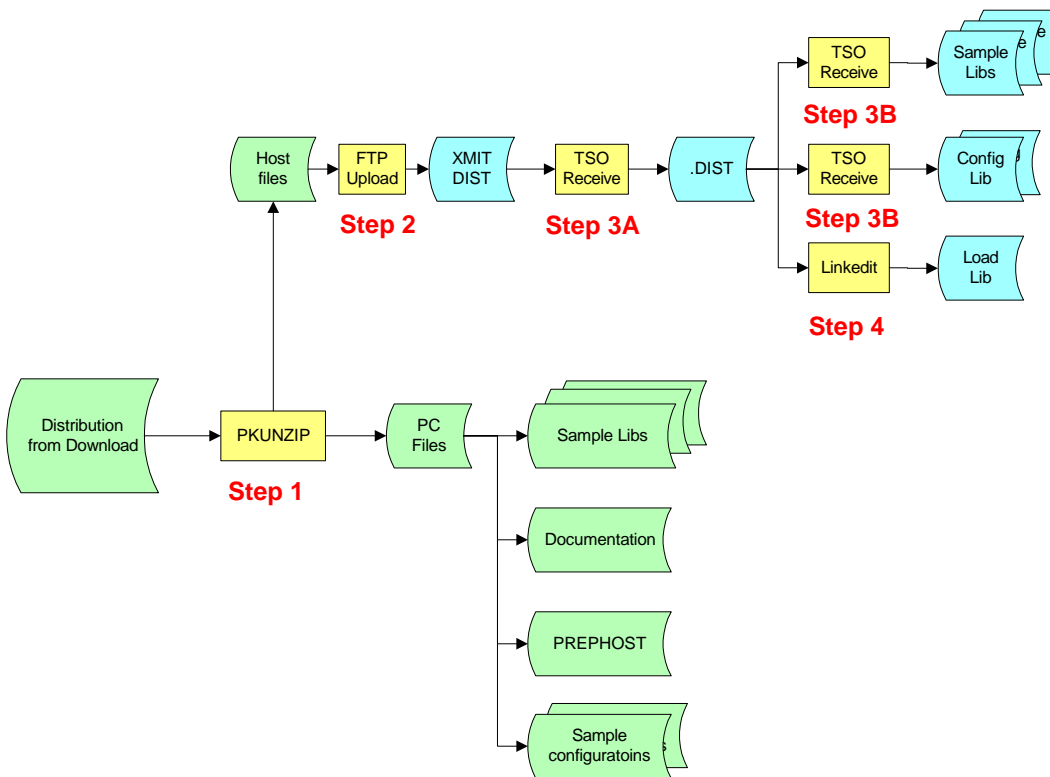
Installation Task list:

1. Unzip distribution file on PC
2. Transfer XMITDIST and RCVINST file to host
3. Unload (TSO Receive) XMITDIST in DIST library
(optional) Unload (TSO Receive) sample libraries
Unload (TSO Receive) sample configuration libraries
4. Linkedit program
5. Tailor and run installation verification job
6. (For 'old' MVS only) Assemble JESTABLE
7. (Optional) Set up a PROC for batch mode operations
8. Tailor your CONFIG file with authorization codes

Product update.

If you already have the program installed and running, you only need steps 1, 2, 3, and 4.

Here is a graphic representation to accompany the following steps.



Step 1: PKUNZIP the Distribution 'PC' File

- ☐ **Execute *JES2XXX.EXE*** to unzip the library members; you will be given the opportunity to choose the output location.

The distribution consists of a binary host file, **JES2XXX-XMITDIST.BIN** that will be uploaded to the host in the next step, and various PC specific files and sub-directories.

In the main directory into which you unzipped the distribution package, you should find the following:

Folders

CONFFTP	Sample configuration files for JES2FTP
CONFMAIL	Sample configuration files for JES2Mail
DOC	Supplied documentation
SAMPBIN	Sample Binary files as per User Guide
SAMPLIB	Sample JCL and files as per User Guide
SAMPRPT	Sample reports as per User Guide

Files

JES2XXX-XMITDIST.BIN	XMIT file to upload to host
----------------------	-----------------------------

PrepHost.exe Overlay conversion tool, PC program.

Step 2: Transfer XMITDIST and RCVINST files to host.

The allocate space for the uploaded file

- ☐ Allocate a file for uploading, in this example we call this file `HLQ.JES2XXX.XMITDIST`. The dataset attributes are:

Record Length:	80	
Block Size:	27920	(As appropriate)
Organization:	PS	(Sequential)
Allocate by:	Blocks	
Allocation:	336	

The uploaded file will be about 117,000 records (9.0 meg).

The Upload XMITDIST and RCVINST files.

The `JES2XXX-XMITDIST.BIN` distribution file is the main product file, the `RCVINST.JCL` file is a sample job for unwrapping it.

- ☐ **Upload the XMITDIST files to the Host file** Since the file is coded in EBCDIC it must be transferred in “binary” mode. The `RCVINST` file is text and should be uploaded as a normal text file with translation.

Use either `IND$FILE` or an FTP client for the transfer. For example, the following commands issued to the Windows FTP client will transfer the files to an MVS FTP host and will result in these responses.

The `RCVINST` file can be uploaded directly to a library for JCL, or the FTP server will automatically allocate space for it, as it does here.

```
C: \
ftp hostname or IP address of host
220 Welcome to xxxx
User: USERID
331 Send password please
Password: xxxxxxxx
230 USERID is logged on. Working directory is "USERID."
ftp> LCD distribution.foldername
Local Directory is now ...
ftp> Put RCVINST.JCL 'hlq.JES2XXX.RCVINST.jcl'
200 Port request ok
125 Storing dataset HQL.JES2XXX.RCVINST.JCL
FTP: 3552 bytes sent in ...
ftp> BINARY
200 Representation type is Image
ftp> Put JES2XXX-XMITDIST.BIN 'HLQ.JES2XXX.XMITDIST'
200 Port request ok
125 Storing data set "HLQ.JES2XXX.XMITDIST"
ftp> 9389920 bytes sent in ...
ftp> Quit
C: \
```

Step 3: Use RCVINST job to unload the libraries

- This step will use the TSO RECEIVE command to expand the XMTIDIST uploaded file to a ..DIST PDS (Library).

The steps in the job are:

1. (Optional) Delete the libraries
2. Allocate PDSs for the various libraries including a LOADLIB
3. Receive the ..DIST library from the XMITDIST file
4. Receive the sample libraries from the ..DIST library

- This job was must be tailored to your site:

```
//RCVINST JOB ...
//*
//* -----
//* Delete existing libraries - If needed
//* -----
//*
//*STEPDEL EXEC PGM=IEFBR14, COND=(0, LT)
//*DELE1 DD DSN=hl q. JES2XXX. DIST, DISP=(MOD, DELETE),
//*        SPACE=(TRK, (0)), UNIT=3390
//*DELE2 DD DSN=hl q. JES2XXX. SAMPBIN, DISP=(MOD, DELETE),
//*        SPACE=(TRK, (0)), UNIT=3390
//*DELE3 DD DSN=hl q. JES2XXX. SAMPLIB, DISP=(MOD, DELETE),
//*        SPACE=(TRK, (0)), UNIT=3390
//*DELE4 DD DSN=hl q. JES2XXX. SAMPRPT, DISP=(MOD, DELETE),
//*        SPACE=(TRK, (0)), UNIT=3390
//*DELE5 DD DSN=hl q. JES2XXX. CONFFTP, DISP=(MOD, DELETE),
//*        SPACE=(TRK, (0)), UNIT=3390
//*DELE6 DD DSN=hl q. JES2XXX. CONFMAIL, DISP=(MOD, DELETE),
//*        SPACE=(TRK, (0)), UNIT=3390
//*DELE7 DD DSN=hl q. JES2XXX. LOADLIB, DISP=(MOD, DELETE),
//*        SPACE=(TRK, (0)), UNIT=3390
//*
//* -----
//* Allocate libraries as PDS
//* -----
//*
//ALLODSN EXEC PGM=IEFBR14, COND=(0, LT)
//DSN1 DD DSN=hl q. JES2XXX. DIST, DISP=(, CATLG, DELETE),
//        DCB=(RECFM=FB, LRECL=80, BLKSIZE=4000), UNIT=SYSALLDA,
//        SPACE=(4000, (2000, 500, 30))
//DSN2 DD DSN=hl q. JES2XXX. SAMPLIB, DISP=(, CATLG, DELETE),
//        DCB=(RECFM=FB, LRECL=80, BLKSIZE=8000), UNIT=SYSALLDA,
//        SPACE=(8000, (50, 25, 30))
//DSN3 DD DSN=hl q. JES2XXX. SAMPBIN, DISP=(, CATLG, DELETE),
//        DCB=(RECFM=U, LRECL=0, BLKSIZE=27998), UNIT=SYSALLDA,
//        SPACE=(27998, (10, 1, 30))
//DSN4 DD DSN=hl q. JES2XXX. SAMPRPT, DISP=(, CATLG, DELETE),
//        DCB=(RECFM=FBA, LRECL=133, BLKSIZE=13300), UNIT=SYSALLDA,
//        SPACE=(13300, (10, 1, 30))
//DSN5 DD DSN=hl q. JES2XXX. CONFFTP, DISP=(, CATLG, DELETE),
//        DCB=(RECFM=FB, LRECL=80, BLKSIZE=4000), UNIT=SYSALLDA,
//        SPACE=(4000, (10, 2, 30))
//DSN6 DD DSN=hl q. JES2XXX. CONFMAIL, DISP=(, CATLG, DELETE),
//        DCB=(RECFM=FB, LRECL=80, BLKSIZE=4000), UNIT=SYSALLDA,
//        SPACE=(4000, (10, 1, 30))
//DSN7 DD DSN=hl q. JES2XXX. LOADLIB, DISP=(, CATLG, DELETE),
//        DCB=(RECFM=U, LRECL=0, BLKSIZE=19069), UNIT=SYSALLDA,
```

```

//          SPACE=(TRK, (200, 50, 30))
//*
//* -----
//* TSO Receive the DIST library from the uploaded XMITDIST file
//* -- Step 3A --
//* -----
//*
//STEP3A EXEC PGM=IKJEFT01, COND=(0, LT)
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
RECEIVE INDSNAME(' hlq.JES2XXX.XMITDIST')
          DSNNAME(' hlq.JES2XXX.DIST') SHR
//*
//* -----
//* Receive the sample libraries from the member of the DIST library
//* -- Step 3B --
//* -----
//*
//STEP3B EXEC PGM=IKJEFT01, COND=(0, LT)
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
RECEIVE INDSNAME(' hlq.JES2XXX.DIST(SAMPLIB)')
          DSNNAME(' hlq.JES2XXX.SAMPLIB') SHR
RECEIVE INDSNAME(' hlq.JES2XXX.DIST(SAMPBIN)')
          DSNNAME(' hlq.JES2XXX.SAMPBIN') SHR
RECEIVE INDSNAME(' hlq.JES2XXX.DIST(SAMPRPT)')
          DSNNAME(' hlq.JES2XXX.SAMPRPT') SHR
RECEIVE INDSNAME(' hlq.JES2XXX.DIST(CONFFTP)')
          DSNNAME(' hlq.JES2XXX.CONFFTP') SHR
RECEIVE INDSNAME(' hlq.JES2XXX.DIST(CONFMAIL)')
          DSNNAME(' hlq.JES2XXX.CONFMAIL') SHR
//

```

When complete, the HLQ.JES2XXX.DIST library (PDS) should contain the following members:

CONFFTP	XMIT of JES2FTP Configuration Library
CONFMAIL	XMIT of JES2Mail Configuration
J2X	Object of JES2xxx
J2XOE	Object of JES2xxx w/OE,
J2XOEMQ	Object of JES2xxx w/OE and MQ
J2XPRES	Prelink Object of JES2xxx
SAMPBIN	XMIT of SAMPBIN library
SAMPLIB	XMIT of SAMPLIB library
SAMPRPT	XMIT of SAMPRPT library

After the second RECEIVE is completed, it will unpack the sample libraries:

SAMPLIB	JCL for Samples
SAMPRPT	Sample report files
SAMPBIN	Sample binary files
CONFMAIL	Sample JES2Mail configuration library
CONFFTP	Sample JES2FTP configuration library

For JES2Mail you should find the following files in the **..CONFMAIL** library:

AGENTS	Sample 'table' for SCRIPT2
ASMJEST	JCL to assemble JESTABLE
BLUEBAR	Sample overlay definition
CONFIG	Sample configuration file

COVER	Sample non-HTML 'cover page'
COVERHTM	Sample HTML 'cover page'
DEFAULT	Sample RuleSet for default routing
GREENBAR	Sample overlay definition
IRSW2	Sample overlay form for IRS W2
IVPMail	Sample Installation and Verification Job
JES2IN	SYSIN commands for JES2PROC
JES2PROC	Sample of started procedure JCL
JESTABLE	Offsets for older JES API
LINKJ2M	Linkedit JCL for 'standard' version
LINKJ2MO	LinkEdit JCL for OE version
LINKJ2MQ	LinkEdit JCL for OE + MQ VERSION
LINKPRE	PreLinkedit and Linkedit for 'standard' version
SALES	Sample RuleSet for a "sales report"
SCRIPT1	Sample Report Splitter script
SCRIPT2	Sample Report Splitter script

- It is recommended that you create your installation `..CONFIG` library and copy and tailor the files from the `CONFMAIL` library as needed. All of the examples in this manual will refer to the "`CONFIG Library`" as your working library. Within this library is the "`CONFIG File`" that contains the system configuration statements.

Step 4: Linkedit JES2Mail

Supplied object modules

The executable module is supplied in four variations, you only need to linkedit one, but you are free to linkedit them all. The object files are:

J2X	Standard 'old MVS' version.
J2XOE	Open Edition (Unix system Services) version – Recommended
J2XOEMQ	Open Edition and MQ Series support
J2XPRES	Prelink version for use with CA/Interlink TCP/IP stack

Linkedit JES2MAIL

Open Edition API

You will use ...`CONFMAIL` distribution library member `LINKJ2MO`

- **Modify and run the `LINKJ2MO` job** to local standards and execute.

This will linkedit the `J2XOE` object module into the `JES2MAIL` load module.

Note: Since the module must run in an authorized state, the linkedit option of `AC(1)` is required.

Step 5: Run the Installation Verification Job

The following is a skeleton for the supplied installation verification job. It is supplied in the **CONFMAIL** distribution library as member **IVPMAIL**. Items to configure are highlighted.

```
//IVPBATCH JOB 1,IVP -> Adjust this JOB statement!
//*
//* *-----*
//*
//* This job will test the JES2Mail system.
//* This "batch" (non-authorized) version will
//* deliver a test report to an e-mail server you
//* designate. The job *does not* depend on a
//* configuration file, it is included inline.
//* A second deliverable is a simple message
//*
//* There are several areas of configuration needed:
//*
//* (1) JCL ... including the JOB statement, above!
//* (2) Authorization codes
//* (3) TCP/IP parameters
//* (4) Target e-mail address for messages
//*
//* *-----*
//*
//* *-----*
//*
//* (1)
//* Here configure the job for your installation
//* using the following JCL "SET" statements (if
//* supported by your version of MVS).
//*
//* *-----*
// SET PGM=JES2MAIL           The JES2Mail program
// SET STEPLIB='SYS2.LINKLIB'  The load library containing it
// SET LE370LIB='CEE.SCEERUN'  The LE runtime library
// SET CONFIGLB='hlq.jes2mail.config' JES2Mail configuration PDS(e)
//
//
// EXEC PGM=&PGM, REGION=OM, PARM='/DD:IVPCNFIG'
//STEPLIB DD DSN=&STEPLIB,DISP=SHR
// DD DSN=&LE370LIB,DISP=SHR
//CONFIG DD DSN=&CONFIGLB,DISP=SHR
//LOGFILE DD SYSOUT=*
//IVPCNFIG DD *
;
; *-----*
;
; (2)
; Here code your program authorization codes
;
; *-----*
AuthCode:          0000-0000-000
SplitAuthCode:    0000-0000-000
OverlayAuthCode:  0000-0000-000
;
; *-----*
;
; (3)
; Here code your installation TCP/IP parameters
;
; *-----*
OurDomainName:    hostname.domain.com
PostOffice:       mail.domain.com
PODown:           No post office's are reachable, call x123
LocalZone:        -0800 [PST]
;
/*
//SYSIN DD *
```

```

TRACE=FULL
Formname=IVP
Ruleset=DD:IVPRULES
Report=DD:IVPREPRT ForceASA
File=DD:IVPMSG
END
/*
//IVPRULES DD *
ID: IVP
Name: Installation Verification Procedures
To: xxxxxx@yyyyyyyyyyyy ; (4) e-mail address
From: &Userid;@&OurDomainName;
PackageType: PDF
FontSize: 9
Font: Courier
PageHeight: 8.5
PageWidth: 11
Action: Mail
Recognizeby: Formname: IVP
/*
//IVPREPRT DD *
1JES2MAIL IVP TEST REPORT - HEADER LINE - Page 1
  Line 2
  Line 3 ... Last
1JES2MAIL IVP TEST REPORT - HEADER LINE - Page 2
  Line 2
  Line 3 ... Last
/*
//IVPMSG DD *
To: xxxxxx@yyyyyyyyyyyy ; (4) e-mail address
From: &ZUserid;@&ZOurDomainName;
Subject: Installation Test of Simple Message
Date:

This is a simple message.
.

```

This job is an ‘all-in-one’ demonstration. Generally, the system configuration (IVPCNFG), RuleSets (IVPRULES), and other files are loaded from the PDS(E) specified in the CONFIG DD statement.

Assuming that the CONFIGLB variable, or the CONFIG DD statement points at the PDS(E) where you uploaded all of the sample files, you might try adding the following statement to the RuleSet, just after the PackageType statement is ok:

```
Overlay: GreenBar
```

This should generate the report with a traditional ‘green bar’ overlay. (Look in the file “GreenBar” to see the overlay specifications).

In this example, we have the report ‘Inline’ also, to force it to be treated with ASA carriage control, the “ForceASA” parameter of the REPORT= command is set. Normally, the system determines the carriage control type from the dataset attributes itself or JES.

The result of the job should be two e-mails. The first will contain an attached PDF file, the second is a basic message.

Please note that the last line in the IVPMSG file, the simple message, is a period (.) by itself. This and forgetting a blank line after the message heading are the two most common mistakes when composing simple messages.

Step 6: Optionally Write Batch Mode Procedures

You may wish to simplify program users' work by providing JCL procedures for batch mode operation. (In batch mode, the program is executed within a job to process a particular disk file).

- ❑ **Write the procedure.** The following is a sample procedure to execute the program in batch mode. It should be placed in a suitable PROCLIB.

Customize the procedure with your own dataset names.

```
//JES2MAIL PROC  
//JES2MAIL EXEC PGM=J2MOE, PARM="/CONFIG1"  
//LOGFILE DD SYSOUT=*  
//CONFIG DD DSN=USER.JES2MAIL.CONFIG, DISP=SHR  
//STEPLIB DD DSN=USER.JES2MAIL.LOAD, DISP=SHR  
// DD DSN=CEE.SCEERUN, DISP=SHR
```

The program name will be J2MOE for the version that uses Open Edition, JES2MAIL when other stacks are used. The PARM specifies that the CONFIG1 member of the configuration (CONFIG DD) dataset will define the operating environment.

LOGFILE is a print file that is used for all levels of tracing and logging.

The CONFIG DD statement must point to a PDS or PDSE that will contain all of the configuration and RuleSet files. This *could* be the distribution PDS, but that is not recommended.

The STEPLIB concatenates the load library where the program is located and the Language Environment runtime library. *In batch mode the libraries do not have to be APF-authorized.*

The PARM on the EXEC statement can select a different configuration file set to use. If the PARM is omitted then the default member name of CONFIG is used. When a C++ program is executed, any options in the PARM before the slash (/) are processed by the C++ runtime. The options after the slash are passed to the executing program, so don't forget the slash if you want to use some configuration file other than CONFIG.

If you have multiple TCP/IP stacks and have to select a particular one for the program, see the *Multiple TCP/IP Stacks* note, below.

Step 7: Create an Initial "Configuration File" Member

A "Configuration File" member of the CONFIG DD PDS defines the program's operating environment during the current invocation. A configuration file is selected using the JCL PARM= on the EXEC statement for the program.

You may build one or more "Configuration File" members and place them in a PDS. Initially, you may use the distribution/configuration PDS created above but this is not recommended for long term use.

- An entire chapter, *System Configuration File*, is devoted to this topic.

The absolute minimum configuration file member consists of just the authorization codes provided to run the program:

```
Authcode: 12345678901
```

You will also likely code a **Ruleset:** statement; this statement selects the configuration member where routing and processing directives may be found (those directives are described in *Controlling Report Processing*).

Additional statements may be provided, but they largely specify defaults. For example, if output is directed to a customary host, its name may be placed here rather than in each "RuleSet" file.

Execution Prerequisites

The JES2MAIL program requires the Language Environment and supports the IBM TCP/IP stack.

The Batch Job mode of operation does *not* require execution from an APF authorized library.

RACF Authorization

RACF authority is needed for JES spool operations, but is usually is not needed for Batch operations.

3

System Configuration

The configuration file contains information about the startup of the program. The following example will wait for work based on the JES Writer Name of "JES2MAIL", and the DESTID attribute of the report will be used for a Report RuleSet member name.

```
OurDomainName: casisoft.com
PostOffice: smtpmail.casisoft.com
Trace: STD
Local Zone: -0700 [PDT]
AuthCode: 12345678901
```

The standard member name for the configuration file is "CONFIG". This may be changed by the PARM value on the EXEC statement. Refer to the sections on Batch or JES Writer execution for more information.

Configuration Statements

All configuration statements consist of a keyword, a colon (or equal sign) and a parameter. The keyword must start in the first column of the record, and be immediately followed by the colon or equal sign. The parameter should follow the colon by one or more spaces.

Keywords and most parameters can be in upper or lower case.

If the line begins with a semicolon (" ; ") or asterisk (" * ") it is ignored as a comment.

/INSERT instruction

The **/INSERT** is used to insert a file into the configuration processes.

The `/INSERT` must begin in the first column of the statement, and is followed by a filename. If the filename is less than nine characters in length, it is assumed to be a member of the CONFIG PDF. If the filename begins with “`DD:`”, then it is a reference to a JCL DD statement; otherwise the filename must be a fully qualified MVS dataset name.

`/INSERT` instructions can be nested 10 deep; that is, an “inserted” file can contain `/INSERT` instructions.

For example:

```
/Insert AuthCode
```

AuthCode: (Required)

The vendor of JES2MAIL supplies an authorization code. It contains an expiration date in an encrypted form. Multiple `AuthCode` statements may be present for multiple CPU licenses, such as in a SysPlex environment.

The authorization code is an eleven-digit number. For example:

```
AuthCode: 12345678901
```

For readability purposes the authorization codes can also be entered with imbedded dashes, as in:

```
AuthCode: 1234-5678-901
```

JES2Mail-Lite.

A lite version of JES2Mail can be authorized with the following code:

```
AuthCode: Li te
```

Even if you are licensed for JES2FTP, you can enable the liteware mode of JES2Mail with the above statement.

LocalCodePage:

Specify the local code page for all text files. The internal code page is IBM-1047 (IBM Open Systems). If your installation uses a different codepage, specify it here. Refer to IBM documentation for a complete description, but here are some examples:

```
Local CodePage: IBM-273      ← Germany
Local CodePage: IBM-297      ← France
Local CodePage: IBM-284      ← Spain
```

Note: This command takes effect immediately, so it should be placed near the top of the configuration file.

The IBM-1047 codepage supports Latin-1 character sets. This means that local code pages that are not based on Latin-1 characters, will most likely not be usable. This includes such single byte character sets as Cyrillic and Turkish, and all double-byte character sets.

LocalZone:

The **LocalZone** is used when the system generates a Date: for the e-mail message. The format should be "+" or "-" from GMT. Optionally, a text description can be added, such as "PDT", enclosed in 'square brackets' ("[]"). The default is "-0700 [PDT]". For example:

```
Local Zone: -0800 [PST]
Local Zone: -0500 [EST]
```

OurDomainName: (Required)

When connecting to an SMTP post office, the caller must identify their domain name. This is supplied here. This is generally the 'Host Name' of the machine, but it could be your company 'Domain Name'. For example:

```
OurDomainName: casisoft.com
OurDomainName: host1.company.com
```

OverlayAuthCode: (Required for Form Overlay)

This code will authorize the use of the Overlay feature.

The authorization code is an eleven-digit number: Multiple **OverlayAuthCode** statements may be present for multiple CPU licenses, such as in a SysPlex environment. For example:

```
OverlayAuthCode: 12345678901
```

PODown:

If no connection to any post office can be established and the **RetryLimit** is exceeded, what message text should be sent to the operator console to alert them of the situation? The default is:

```
"J2M001 JES2Mail cannot access any e-mail PostOffice".
```

If you want something else, supply it here. For example:

```
PODown: Call Joe at x234 that PO is down.
```

If you don't want a message to go to the operator console, since it is displayed such that the operator must manually delete it, supply a null message such as:

```
PODown:
```

PostOffice: (Required)

The target post office host address must be supplied. This is the machine name of an SMTP-compliant e-mail system that is to handle all of the outgoing messages.

This *must* be a specific machine name or IP address. It cannot be a name that resolves to multiple addresses. Generally, this is either an e-mail server on your site or at your ISP. If a name is used instead of an IP address, then the TCP/IP Domain Name Service must be available.

For example:

```
PostOffice: e-mail1.company.com
```

Up to ten post offices can be specified by including additional PostOffice statements. If there is a problem with one, JES2MAIL will try the next. Also, in a HeartBeat operation, all post offices can be checked for accessibility.

SetEnv:

This statement can be used to set the environment variables that C/C++ and LE use for various functions. To set a variable, follow the **SETENV** command with the variable name, a space, and the value desired.

Most likely, the only variable that you should change is the **TZ** variable, used in a POSIX (open edition) environment to control Time Zone computations.

To set the **TZ** variable to the Pacific time zone, use a statement like:

```
SetEnv TZ PST8PDT
```

The **TZ** variable values are described in the “Using the TZ or _TZ Environment Variable to Specify Time Zone” section of the “C/C++ Programming Guide”.

Contact product support for help in setting these values.

SplitAuthCode: (Required for Report Splitter)

This is the authorization code for the Report Splitter option of JES2MAIL. It contains an expiration date in an encrypted form. Multiple **splitAuthCode** statements may be present for multiple CPU licenses, such as in a SysPlex environment.

The authorization code is an eleven-digit number. For example:

```
SplitAuthCode: 12345678901
```

Trace:

The system has extensive trace capabilities. The starting ‘trace’ level can be set in the configuration file, but it can also be changed by sending a trace command to the started task with an MVS **MODIFY** operator command (“F taskname,trace=xxxx”).

The trace levels are **MIN**, **STD**, **FULL**, **DUMP**, and **DEBUG**. **MIN** (minimum) and **STD** (standard) produce a small LogFile. Upping the level to **FULL** can get quite voluminous, and **DUMP** is extremely detailed. The **DEBUG** level is mostly used in the RepSplit Script process.

The **FULL**, **DUMP** and **DEBUG** levels are for diagnostic purposes.

The default is **STD** for standard. For example:

```
Trace: Full  
Trace=FULL
```

From an operator command:

```
F JES2MAIL, TRACE=FULL
```

ZipArchiveSize:

With ZIP390 and now native zipping, the archive (ZIP file) is built in memory. As such, we will dynamically “getmain” the specified amount of memory to hold the entire ZIP archive/file. The storage is specified in bytes. The storage is not acquired until the first attachment request for the ZIP operation, and it is released after the attachments have been zipped and shipped. The default is 2 meg. Example:

```
Zi pArchi veSi ze: 4000000 ; 4 meg
```

See chapter on ZIP support for more information.

ZIPService:

Specify the ZIP server that is available. The valid values are “ZIP390”, “PKZIP” and “NATIVE”. If no ZIP service is desired, this statement should not be present, or specify “No”. Example:

```
Zi pServi ce: NATI VE
```

See chapter on ZIP support for more information.

ZipWorkBufferSize:

The ZIP390 AND NATIVEZIP compression API works more efficiently when the input data is supplied in as large a buffer as possible. To accommodate this, the **ZIPWorkBufferSize** should be specified large enough to hold the entire original source file. However, it should also work with smaller values. The default size is 1 meg. Example:

```
Zi pWorkBufferSi ze: 2000000
```

See chapter on ZIP support for more information.

4

Batch Job Mode

Overview

Batch mode execution has two general purposes. The first is to send a text file that contains message addressing as well as the message body (Message/File Mode). The second is to process a report file that is stored in a sequential dataset (Report Mode).

The Message/File mode processes a sequential file that contains inline addressing and message content. The file could be a program generated file and contain any number of messages, or a manually created dataset. Message/File mode messages are not formatted into PDF or HTML. They may specify attachment files, which JES2MAIL will incorporate into the e-mail. Both text and binary attachments are supported.

In REPORT mode, a file called the RULESET supplies the e-mail's addressing and other formatting attributes. REPORT mode provides not only the ability to 'package and ship' a report, but can invoke a script to 'split' the report into sub-reports, and route each individually. A further REPORT mode can overlay a preformatted 'form' to be 'printed on'. An overlay can be a simple 'Green Bar' background, or an elaborate form designed with Adobe Acrobat or PageMaker.

Refer to the Control Reports with RuleSets section for information about RuleSets, and System configuration for details about the configuration file.

JCL

The JCL to execute the program in batch can look like this:

```
//JMAIL JOB ...  
//*  
//JES2MAIL EXEC PGM=JES2MAIL, PARM=' /CONFIG1'
```

```

//LOGFILE DD SYSOUT=*
//CONFIG DD DSN=USER.JES2MAIL.CONFIG,DISP=SHR
//STEPLIB DD DSN=USER.JES2MAIL.LOAD,DISP=SHR
// DD DSN=SYS1.SCEERUN,DISP=SHR
//SYSIN DD *
commands

```

LOGFILE is a print file that is used for all levels of tracing and logging.

The CONFIG DD statement must point to a PDS or PDSE that will contain the "CONFIG1" member referenced in the PARM of the EXEC statement.

The STEPLIB includes the Load library where JES2MAIL is located. The SCREERUN DD may not be required in your installation. It points to the Language Environment runtime library.

SYSIN is read for command input.

The PARM on the EXEC statement can select a different 'Configuration' file to use. If the PARM is omitted, then the default member name of "CONFIG" is used. When a C++ program is executed, any 'options' in the PARM before the slash (/) are processed by the C++ runtime. The options after the slash are passed to the executing program. So to provide a different configuration file to load, the PARM would be **PARM= '/CONFIGX'**.

Command Input

FILE command

The FILE statement specifies the sequential dataset that contains the inline addressed e-mail messages to process and send. As in:

```

FILE=MESSAGE.FILE.PDS(MSG1)
file=dd:message

```

Refer to the *Simple Message File mode* section for additional information.

TRACE command

The **TRACE** command will override the **TRACE** statement from the CONFIG file. Valid trace levels are **MIN**, **STD**, **FULL**, **DUMP** or **DEBUG**. As in:

```
TRACE=FULL
```

REPORT command

The Report mode is used when the report is not in JES, but is in a sequential dataset.

The report/file should have ASA controls. If not, the report is processed as a simple sequential file without any pagination, unless the "ForceAsa" option is used.

REPORT= uses the same Report RuleSet configuration as the started task mode, and the RuleSets can include Report Splitting Scripts, and Overlay definitions.

```
TRACE=FULL
RULESET=BOBNT
REPORT=USER. PRINT. PDS(PRINT2)
END
```

The Report/File *must* use ASA or MCC carriage control, unless the “ForceASA” option is specified, as in:

```
Report=dd:Repi n ForceASA
```

Wildcard Filenames

If the source report file is in a PDS or PDSE, then a wildcard character can be used in the membername. Two characters have special meaning, the asterisk (*) to specify a partial membername, and a question mark (?) to ‘wildcard’ a single character.

For example:

```
Report=hl q. pri nt. pds(ABC*)
Report=hl q. pri nt. pds(?BC)
```

When the list of files is generated, it is treated as if an individual ‘Report=’ statement were issued for each file.

RULESET command

The **RULESET** command is used in conjunction with the **REPORT=** statement to ‘force’ the RuleSet file to open and use it for report recognition and routing. As in:

```
RULESET=BOBNT
```

FORMNAME command

The **FORMNAME** command is used in conjunction with the **REPORT=** statement to pass a *FormName* to be used in the report recognition and routing process, as in:

```
TRACE=FULL
RULESET=BOBNT
FORMNAME=STOCK1
REPORT=USER. PRINT. PDS(PRINT2)
END
```

Examples:

Send a report in Batch mode

Here is an example of a job that contains commands and the message.

```
//JMAIL1 JOB ...
//*
//JES2MAIL EXEC PGM=JES2MAIL
//LOGFILE DD SYSOUT=*
//CONFIG DD DSN=USER. JES2MAIL. CONFIG, DISP=SHR
//STEPLIB DD DSN=USER. JES2MAIL. LOAD, DISP=SHR
// DD DSN=SYS1. SCEERUN, DISP=SHR
//SYSIN DD *
TRACE=FULL
RULESET=BOBNT
REPORT=USER. PRINT. PDS(PRINT2)
```

END

In this example the “USER.PRINT.PDS(PRINT2)” contains the report. The file should be created with an FBA attribute so that the first column contains ASA control characters.

The Report Rules in the “BOBNT” member of the CONFIG dataset will determine the routing.

Send a report with inline RuleSet

This example will include the Report Routing RuleSet within the jobstream.

```
//JES2MAIL EXEC PGM=JES2MAIL
//LOGFILE DD SYSOUT=*
//CONFIG DD DSN=USER.JES2MAIL.CONFIG,DISP=SHR
//STEPLIB DD DSN=USER.JES2MAIL.LOAD,DISP=SHR
// DD DSN=SYS1.SCEERUN,DISP=SHR
//SYSIN DD *
TRACE=std
RULESET=DD:RULE ← Point at RuleSet
REPORT=USER.PRINT.PDS(PRINT1)
END
//RULE DD * ← RuleSet is inline
NAME: REPORT TEST
TO: ROBERT LABAYNE <USER@DOMAIN.COM>
FROM: JES2MAIL@CASISOFT.COM
PACKAGETYPE: PDF
FontSize: Medium
```

5

Simple Message File Mode

Overview

A Message File is a sequential file that contains both addressing and message text. This is *not* a report. The messages are simple text or HTML messages. Attachments can be included with the message.

The Message File can be fixed or variable length. Records can be as long as 256 characters. The message file can be either a sequential dataset or a JES Queue dataset. A file can contain any number of messages.

The format for a JES2MAIL Message consists of:

- Message header statements
- Blank line
- Message text
- Period line

SMTP Message Header statements

The SMTP standards provide for a large number of 'header statements'. At a minimum, a message should have a To, From, Subject, and Date. Some e-mail systems are more 'picky' than others, but if you supply at least these four, the message should be acceptable.

For a full discussion of SMTP header statements, refer to the Internet RFCs that discuss message format and MIME encoding.

The JES2MAIL program reads all of the header lines for a message, and then parses out the "To:", "CC:", "BCC:", and "From:" statements to instruct the post office server on message recipients and return addressing. After the post office server has accepted the "From:" and at least one of the recipient addresses, then all of the header lines are sent untouched, except for the **ATTACH:** statement.

If in the processing of the message header the program finds an "**ATTACH:**" statement, then the program prepares to attach the specified file. The program will then insert the proper MIME header statements for multi-segment file encapsulation. The **ATTACH:** statement itself is not passed through.

Message/File Header Statements

FROM:

Specify a return e-mail address. Example:

```
FROM: Sal es@company. com
```

To:

Specify a primary e-mail address. Multiple addresses can be supplied with either multiple statements, or by separating the e-mail addresses by a semicolon (;) or comma (,). Example:

```
To: Gtamas@company. com  
To: @LI ST=USER. LI ST. FI LE  
To: user1@domai n1; user2@domai n2  
To: Sal l y Jones <sal l y@company. com>  
To: Bob LaBayne
```

In the case of the "Bob LaBayne" entry, since it is not a fully formed e-mail address (no '@'), then it will be checked against the DefaultAddressTable, if one was defined.

CC:, BCC:

Specify secondary e-mail addresses.

Date:

If the "**DATE:**" statement is entered with nothing after it, then the program will generate a properly formatted SMTP compliant date for the local date and time. Example:

```
Date:
```

will insert:

```
Date: Mon, 26 Jul 1999 08:30:29 -0700 [PDT]
```

Subject:

This statement is used to supply the message with a subject. Example:

```
Subject: Thi s i s a t e s t m e s s a g e
```

Content-Type:

The content type is *only* required if the message is formatted in HTML. This statement must conform to MIME standards. To set the message as an HTML message, supply a statement such as:

```
Content-Type: text/html
```

Attach:

The **ATTACH:** statement is followed by the filename and various options.

```
TYPE: file-type  
AS: new-filename  
NOTRim  
ZIP  
PASS: password  
ZipClose  
ZipName: archive-name
```

Source-name The filename to be attached immediately follows the ATTACH statement. The filename must be a fully qualified filename, or a DD: reference. If the filename specifies a PDS(E), the member name can be a wildcard of asterisk (*) for multiple characters or question mark (?) for a single character. When a wildcard is detected, it is expanded and treated as an individual ATTACH statement for each qualifying filename.

File-Type. If the filetype is "text", "txt", "html", or "htm", then the file is treated as TEXT, and will be translated from EBCDIC to ASCII. Any other value will force the file to be treated as BINARY, and will be attached as is, no translation.

New-filename can specify a 'PC friendly' filename for the attachment. When this is not specified, then the source filename is used. If the source filename specifies a PDS(E) member, you can use a special symbolic, &member; to reference the member name.

For example:

```
Attach: hlq.file.pds(Report1) as: &Member;.txt
```

NOTRIM. Normally, when TEXT files are processed, trailing spaces are removed from the record before the ASCII new-line character is appended to each line. If you want to suppress the trimming, add the **NOTRIM** option to the statement.

ZIP. If a zipping service has been specified in the configuration file, the file is ZIP compressed into a .ZIP archive file (See section on zipping attachments).

zipClose When zipping attachments, the norm is to add them all to a single ZIP archive file for delivery. If you want to close and ship a ZIP archive explicitly, then specify "ZIPClose" on an ATTACH statement by itself. It can be followed by other ATTACH statements with zipping, but they will start a new archive.

ZipName: name The default ZIP archive name, that is the name that the recipient will see as the attachment's filename, is "Archive.ZIP". To specify an archive name, include this parameter followed by the new name on the first ATTACH statement for the archive.

ATTACH statements can be continued by placing a dash at the end of the line, and continuing on the next line.

Examples:

```
Attach: USER.FILE.SOURCE(TEXT1) Type: text
Attach: USER.OBJECT.PDS(OBJ1) Type: binary
Attach: ABC.FILE Type: Binary As File.bin
Attach: ABC.Longfilename.souce(text) -
Type: Text As: "Long PC filename.txt"
Attach: user.filename Type: text -
as: 'PCFile.txt' zip
Attach: hlq.filename.pds(j2x*) type: text as: &membername;.jcl -
Zip
```

Insert additional files

Additional text can be inserted into the message input with an **/INSERT** statement. Inserted files can be nested ten levels. For example:

```
/INSERT PDSMEM1
/INSERT hlq.dataset.pds(file1)
```

Symbolic Substitution

As the message file is processed, it is scanned for variables bounded by an ampersand (&) and semicolon (;), as in "&ZjobName;". The variables are fully described in the Report Process Language section of this manual. Not all of the RPL variables will have useful values when processed in this mode; many are only appropriate when processing reports or script language. If the variable is undefined, it will be passed through, with its leading and trailing syntax, to the final message.

The values from globally defined variables, those that are defined in the system configuration with the **Define** statement, can also be inserted.

Here are some of the variables available for insertion.

ZJobName	String, length 8. The current OS/390 Jobname.
ZJobID	String, length 8. The current JobID.
ZProcName	String, length 8. The current OS/390 Procname.
ZStepName	String, length 8. The current OS/390 Stepname.
ZUserID	String, length 8. The current UserID for the job.
ZSysADay	String. The current day as text. For example, "Monday". The description comes from the internal Day of Week table. This can be changed with the DayOfWeekLong configuration statement.

ZSysADayShort	String. The current day as text. For example, "Mon". The description comes from the internal Day of Week table. This can be changed with the DayOfWeekShort configuration statement.
ZSysAMonth	String. The current month as text. For example, "January". The description comes from the internal Month Description table. This can be changed with the " MonthLong " configuration statement.
ZSysDateEDDDMMYYYY	String. The current date in a Day.Month.Year format, i.e. 10.06.2000.
ZSysDateDDMMYY	String. The current date in a Day-Month-Year format, i.e. 10-Jun-00.
ZSysDateMMDDYY	String. The current date in a Month/Day/Year format, i.e. 06/10/00.
ZSysIDay	Int. The current day of the month. Value from 1 to 31.
ZSysIMonth	Int. The current month as an integer. Value from 1 to 12.
ZSysIYear	Int. The current year as an integer value, i.e. 2000.
ZSysTimeHHMMAP	String. The current time in a HH:MM am format.
ZSysTimeHHMMSS	String. The current time in a HH:MM:SS format.
ZSysTimeIHour	Int. Current system hour.
ZsysTimeIMinute	Int. Current system minute.
ZsysTimeISecond	Int. Current system second.

Escape Character

The backslash character is used as an escape character. Any character following the backslash will be passed through with no processing. To send a backslash character, use two, as in "\\".

Word Wrap

Without word wrap, each line of the message file becomes a 'hard' line in the final e-mail. That is, it is terminated with a carriage return and LineFeed (CR/LF). This is simple, and it can result in some awkward presentation by the recipient depending on the width of the e-mail client's presentation window.

By default, for example:

```
This is a  
sentence.
```

will always be displayed as two lines.

With word wrapping, the two lines can be 'wrapped' together, and the recipient's e-mail client determines if they should appear as one or two lines.

Wrap with 'Tag' controls

There are two mechanisms to enable wrapping. The first uses 'tags' to turn on or turn off the wrapping. The tags must appear on a line by themselves and begin in column 1.

For example:

```
<Wrap=on>
This is a
sentence.

This is a second
sentence.
<Wrap=off>
```

To terminate a 'paragraph', a blank line is needed, so even though the above shows a blank line, the presented result will most likely be:

```
This is a sentence.
This is a second sentence.
```

For a 'real' blank line, two must be in the original data.

Zipping Attachments

If you have enabled ZIP compression in the configuration file, you can instruct JES2MAIL to compress the attachments into a ZIP archive file. See the chapter on ZIP Compression Support for more information.

Normally, no matter how many **ATTACH** statements are specified, all of the files are ZIP'ed into a single ZIP file. However the **ATTACH** statement with the "ZipClose" option can be added and then any subsequent **ATTACH** statements will begin a new archive file.

The recipient can unzip the archive files with various tools.

Each individual file within the ZIP archive file can have a password.

To activate the ZIP interface, a few **CONFIG** statements are required. Here are some examples, but there is a more detailed explanation in the configuration chapter.

```
ZipService: Native
ZipArchiveSize: 1000000
ZIPWorkBufferSize: 500000
```

The archive is built in memory, so the **ZIPArchiveSize** must be large enough to hold the entire ZIP file. The **ZIPWorkBufferSize** should be large enough to hold an individual attachment 'source' file.

The storage for the ZIP process is acquired when the first **ATTACH** statement with a ZIP request is processed, and it is all released when all of the attachments have been zipped and shipped.

If the ZIP option is specified, and no ZIP services are available, the file is simply sent without compression.

Here is an example of a job that ZIPs and ships:

```
//JES2MAIL EXEC PGM=J2MOE
//LOGFILE DD SYSOUT=*
//CONFIG DD DSN=USER.JES2MAIL.CONFIG,DISP=SHR
//STEPLIB DD DSN=USER.JES2MAIL.LOAD,DISP=SHR
//SYSIN DD *
FILE=DD: MESSAGE
END
//MESSAGE DD *
to: user@domain.com
date:
subject: Test message zip
Attach: ABC.FILE.ONE type: text -
       as: 'abc.txt' zip
Attach: USER.file.two type: text -
       as: 'user2.txt' zip
Attach: user.binary.data(binfile1) type: bin -
       as: 'binfile.dat' zip

This is a test message. with Zip390.
This is the job: &ZJobName; -&ZJobno;
```

In this example, the final delivered e-mail will contain a single ZIP file attachment, with a name of "ABC.ZIP" (the first filename with a ZIP suffix).

Sample Jobs

Send a Message File with inline addressing

The following is a job to send a file that contains three very simple messages.

```
//JMAIL JOB ...
//*
//JES2MAIL EXEC PGM=JES2MAIL
//LOGFILE DD SYSOUT=*
//CONFIG DD DSN=USER.JES2MAIL.CONFIG,DISP=SHR
//STEPLIB DD DSN=USER.JES2MAIL.LOAD,DISP=SHR
// DD DSN=SYS1.SCEERUN,DISP=SHR
//SYSIN DD *
TRACE=STD
FILE=USER.EMAIL.MSG(TESTIN1)
END
```

Here is the message input file:

```
To: user@domain.com
cc: @list=casi.erpr000.sourcec(list1)
from: <>
date:
subject: Test message 1
                                     ← Blank line
This is the body of the message 1.
.                                     ← End of message
To: Robert LaBayne <user@domain.com>
from: Jerry
date:
subject: Test message 2
                                     ← Blank line
This is the body of the message 2.
.                                     ← End of message
to: @list=casi.erpr000.sourcec(list1)
from: Bob
date:
subject: Test message 3
                                     ← Blank line
This is the body of the message 3.
.                                     ← End of message
```

Send a Message File with attachments

Here is a message input file that specifies some attachments:

```
To: @list=user.list.pds(list1)
From: Bob
Date:
Subject: Test message 3 w/attachment
Attach: user.source.pds(testc2) type: text as: testc2.txt
Attach: user.object.pds(testobj) type: binary as: obj.bin

This is the body of the message.
.
```

This will send the e-mail message, with the two attached files, to every recipient in the distribution list.

All in one JOB Example

Here is an example of a job that contains commands, and the message.

```
//JMAIL1 JOB ...
//*
//JES2MAIL EXEC PGM=JES2MAIL
//LOGFILE DD SYSOUT=*
//CONFIG DD DSN=USER.JES2MAIL.CONFIG,DISP=SHR
//STEPLIB DD DSN=USER.JES2MAIL.LOAD,DISP=SHR
// DD DSN=SYS1.SCEERUN,DISP=SHR
//SYSIN DD *
TRACE=STD
FILE=DD:MESSAGE ← Point at DD statement
END
//MESSAGE DD * ← Message file
to: user@domain.com
from: bob
subject: Test message 1
Attach: USER.FILENAME Type: BINARY As: REP1.BIN
← Blank line
This is the body of the message.
← Last line of message
.
```

Example with word wrap

Here is an example of a job that contains commands, and the message.

```
//JMAIL1 JOB ...
//*
//JES2MAIL EXEC PGM=JES2MAIL
//LOGFILE DD SYSOUT=*
//CONFIG DD DSN=USER.JES2MAIL.CONFIG,DISP=SHR
//STEPLIB DD DSN=USER.JES2MAIL.LOAD,DISP=SHR
// DD DSN=SYS1.SCEERUN,DISP=SHR
//SYSIN DD *
TRACE=STD
FILE=DD:MESSAGE ← Point at DD statement
END
//MESSAGE DD * ← Message file
to: user@domain.com
from: bob
subject: Test message 1
Attach: USER.FILENAME Type: BINARY As: REP1.BIN
← Blank line
<wrap=on>
This is sentence one of the body.
This is sentence two of the body.
This is the second paragraph.
<wrap=off>
← Last line of message
.
```

Example of HTML and variable substitution

The following example will send a message in HTML format. The HTML shown here is the barest minimum of the language's capabilities. Buy one of the thousands of HTML books for complete documentation.

```
//JMAIL JOB ...
//*
//JES2MAIL EXEC PGM=JES2MAIL
//LOGFILE DD SYSOUT=*
//CONFIG DD DSN=USER.JES2MAIL.CONFIG,DISP=SHR
//STEPLIB DD DSN=USER.JES2MAIL.LOAD,DISP=SHR
// DD DSN=SYS1.SCEERUN,DISP=SHR
//SYSIN DD *
TRACE=STD
FILE=DD: MESSAGE ← Point at DD statement
END
//MESSAGE DD * ← Message file
to: user@domain.com
from: bob
date:
subject: Test message 1
Content-Type: text/html
Attach: hlq.datafile.name type:Bin AS: PCFile1.dat
                                     ← Blank line

<html>
<head>
<title>
This is the title
</title>
</head>
<body>
<bold><big>Report variable substitution</big></bold><br>
                                     <br>
<small>Title</small> &lt; &lt;big>Big</big> <br>
                                     <br>
OurDomainName: &zOurDomainName; <br>
                                     <br>
JobName: &zJobname; <br>
JobID: &zJobID; <br>
ProcName: &zProcname; <br>
StepName: &zStepname; <br>
                                     <br>
ZSysDateDDMMYY: &ZSysDateDDMMYY; <br>
ZSysDateMMDDYY: &ZSysDateMMDDYY; <br>
ZSysdateEDMMYYYY: &ZSysDateEDMMYYYY; <br>
ZSysADay: &ZSysADay; <br>
ZSysTimeHHMMAP: &ZsysTimeHHMMAP; <br>
ZSysTimeHHMMSS: &ZSysTimeHHMMSS; <br>
</body>
</html>
. ← Last line of message
```

The most common errors in a message file are:

- Forgetting to put a blank line after the header statements.
- Putting sequence numbers in columns 73-80, don't.

6

Report Processing

Overview

RuleSets are files that are used by JES2MAIL to determine routing and formatting options for the report/file. The RuleSets are usually stored in the CONFIG PDS, but they can be explicitly specified when needed.

There are different approaches to creating the RuleSets. The biggest factor depends on how the target e-mail address is to be determined. One approach is to have a RuleSet for each destination, with the destination e-mail address specified in the RuleSet. Another approach is to have a 'generic' RuleSet, without an e-mail address, and have the e-mail address come from the OUTPUT statement that created the report. A third mechanism that can be used is to have the RuleSet invoke a Report Processing Script, and the script determines the e-mail address.

Each RuleSet file can contain formatting instructions for a number of reports. For example, if the RuleSet is "SALES", and the DestID has specified "SALES", there can be different instructions for different sales reports. Each group of instructions within a RuleSet is called a rule. Rules are also sometimes referred to as Report Descriptions. A blank line separates each rule within the RuleSet. If a report/file matches more than one rule within a RuleSet, it is processed using each matching rule according to the routing and formatting instructions. In this manner a single report might be sent to different destinations with entirely different formatting and packaging.

Here is an example:

```
Name: Default Sales Name
To: GTamas@corpname.com
From: Gtamas@corpname.com
PackageType: PDF
FontSize: 9
```

```

Font: Courier
PageHeight: 11
PageWidth: 8.5

ID: SALES1
Name: Sales Report 1
To: @LIST=CASIDIST.PDS(SALESREP)
RecognizeBy: "Sales Rep Report" Line(1:3) col(40:50)
PackageType: PDF
FontSize: 9
Font: Helvetica
PageHeight: 8.5
PageWidth: 11

ID: SALES2
Name: Sales Report 2
Recognizeby: FormName=SALES2
To: RepToUser@report.com
CC: RepCCUser@report.com
CC: Rep2CCUser@Report.com
To: Rep2ToUser@Report2.com
BCC: RepBCCUser@report.com
PackageType: HTML

ID: STOCK1
Name: Stock Paper 1
recognizeby: formname: Stock1
to: gtamas@intercorp.com
PackageType: pdf
Base64: yes
FontSize: 9
font: Courier
Overlay: GreenBar

```

In the 'SALES1' Report Description, the first page of the report is scanned for the text "Sales Rep Report" from lines one through three and starting at column 40 through 50. If a match is found, it will be formatted and sent to a route list, which we would assume is a list of sales reps.

In the 'SALES2' Report Description, we recognize it by the FormName of "SALES2". In this case, the report is formatted for an e-mail client that can display HTML in the message body. (Most modern e-mail clients do).

In the 'SALES3' Report Description, we should find the text "Daily Sales Report" in the second line at column 35. If a match is found, then the report/file is packaged as a simple TEXT file attachment. It is to be delivered to three e-mail addresses. It is to have a cover sheet (the message body), pulled from the file COVER.SHEET.PDS(COVER1).

In the 'STOCK1' Report Description, we are going to invoke an overlay. Instead of a plain, blank white background, the descriptions in the "GreenBar" file will describe the overlay. Assuming the "GreenBar" file is the supplied sample, a classic green and white bar background will be used, and the text will be printed at a size of 8 lines per inch with 4 lines for each 'bar', something like this:

The Default Rule

The first rule in the RuleSet file is the *default* rule. If the report/file does not qualify for any other rule within the RuleSet, then it is processed and routed according to the statements in the default rule. If the report matches other rules within the RuleSet then

it is processed according to the statements in those rules and then only uses the default rule for certain values not specified explicitly in the matching rule(s).

If a formatting parameter is omitted from a rule, then the appropriate setting from the default rule is used. Thus, if the default rule specifies that the report is to be packaged as a PDF file with a horizontal page format and a font size of 9, then unless the report processing rule specifies otherwise, these default settings are used.

For example, the following RuleSet, called `SALESMGR`, will send all reports with a DestID of "`SALESMGR`" to a specific e-mail address. This RuleSet contains only one rule, which also acts as the default rule for the RuleSet. All of the reports are to be packaged as PDF attached files.

```
Name: SaleManager Reports
To: GBush@corpname.com
PackageType: PDF
FontSize: 9
Font: Courier
PageHeight: 11
PageWidth: 8.5
```

Routing Statements

Each Report Description set can contain any number of routing statements. There can be multiple **To:**, **CC:**, and **BCC:** statements. There should be no more than one **From:** statement.

If no **From:** is specified, then the message is sent with a *null* return address. This is legitimate, but means that the recipient cannot reply to the message.

To:, **CC:**, and **BCC:** addresses can specify either a specific e-mail address or a route list. If the report/file is to be routed to a large number of destinations, then a route list should be used.

When the message is being passed to the post office, it offers each of the recipient addresses to the e-mail server. If no recipient addresses are accepted, the report is skipped, and the process continues with the next RuleSet or report/file. On the other side, if the RuleSet specifies ten destinations and only one is accepted, the report is sent to that one recipient. The post office only does a 'rough' look at the address to determine if it is acceptable. It is very easy for the post office to accept a message for delivery, only to find out later that the final recipient is not a valid address.

Route Lists

Route lists are sequential files that contain an unlimited number of recipient addresses.

FileNames in configuration statements

Several configuration statements use a filename, such as **CoverPage** and **splitScript**.

The filename can be supplied in full, just the membername in the PDS referenced in the CONFIG DD statement, or reference a different DD statement.

Basically, the rule is, if the filename is less than nine characters in length, assume it's a PDS membername; otherwise, use it as a fully qualified name unless it begins with "DD:", in which case the filename should point to a DD statement.

Thus if the CONFIG DD looks like:

```
//CONFIG DD DSN=J2M.CONFIG.PDS, ...
```

and

```
//COVER DD DSN=J2M.CONFIG.PDS(COVER1), ...
```

then the following statements result in the same file:

```
Coverpage: J2M.CONFIG.PDS(COVER1)
Coverpage: COVER1
CoverPage: DD: COVER
```

Delivered Report Formats (PackageType)

Adobe PDF Format

PDF formatting encodes the report according to the Adobe Acrobat "Portable Document Format." In our case, we used the Version 1.2 reference manual dated November 27, 1996.

A PDF formatted document is a highly structured file that allows for rapid navigation using the freely available Adobe Acrobat Reader. The format ensures that the report will be presented the same, no matter what printer or computer is used. PDF documents are always delivered as an attached file with a ".PDF" suffix.

When specifying the fonts that are to be used in formatting reports, remember that the base version of JES2Mail supports only those fonts that are built into Adobe's Acrobat Reader. They are:

Courier	Helvetica	Symbol
Courier-Bold	Helvetica-Bold	Times-Roman
Courier-Oblique	Helvetica-Oblique	Times-Bold
Courier-	Helvetica-	Times-Italic
-BoldOblique	-BoldOblique	Times-BoldItalic

Additional font support is included when the overlay feature is available. Fonts can be 'ripped' from PC PDF files and uploaded to the host for use. (Refer to the PDF Overlay chapter for more information)

The page size is either computed by the *pageheight* and *pagewidth* parameters, or the page orientation.

The default page sizes if only an "orientation" statement is used:

Orientation	Width	Height
Landscape	11	8.5
Portrait	8.5	11

Once the page height is determined, then the approximate number of lines per page is computed according to the font size, with a margin of three lines at the top and bottom

of each page. Thus, if the report/file does not have proper top-of-page formatting, this will be the maximum number of lines per page.

As a starting point, we find that a standard program listing with 132 columns and ASA formatting can fit in the following:

```
PageHeight: 8.5  
PageWidth: 11  
Font: Courier  
FontSize: 9
```

HTML Format

HTML (Hyper Text Markup Language), is usually used for web pages and browser viewing. The SMTP standard specifies support for HTML in the message body. Most modern e-mail clients will support the HTML formatting.

Since the expectation is that a report is being sent, then only a minimum of HTML commands are used.

JES2MAIL supports two HTML formatting options. For standard HTML, JES2MAIL does all of the formatting and will make sure that any HTML characters or triggers in the report/file are 'encapsulated' so that they are presented properly.

Alternatively, a **HTMLASIS** option leaves the HTML formatting to the creator of the report/file. In this case, the e-mail MIME headers will designate HTML as the contents type, and no encapsulation is attempted.

There is one problem with HTML formatting of reports. HTML does not specify a "new page". (Or at least we couldn't find one). The program will insert a horizontal line (<HR>) into the message as a page separator. This can be changed in the RuleSet for the report/file.

Text Format

Report/files can be delivered as attached text files. There are two choices for handling the Carriage Control character. The first is to convert the ASA and MCC controls into CR/LF and FF characters. The second choice is to pass the ASA control characters through to the text attachment.

Rich Text Format (RTF)

Reports can be delivered as attached Rich Text Format (RTF) formatted files. Microsoft Word, Word Pad, and many other PC products support RTF. It provides some of the fidelity control of PDF, without having to install the Adobe Acrobat Reader.

Non-report Data (DATA)

When the input file is not really a report it can be processed as a data file. Data files don't have carriage control at the beginning of each record and they may contain binary data. When data is being processed, instead of reading a 'page' of input and processing it, each record is read and treated as a 'page'. This allows the various script sections to

act on the data file, maybe to extract data, or split a data file into different pieces. This is discussed in more depth in the “Processing non-report data” section of this manual.

ASA and MCC Carriage Control Handling

JES2MAIL can handle reports with ASA format control including the overstrike “+” character. See the RuleSet **MergeOverStrike** statement described later in this chapter for further information.

MCC (i.e., Printer Channel Commands) are handled as best as possible, since some of the 1403/3800 printer commands have no equivalents in an ASA or CR/LF ASCII world. In general, only skip to channel 1 is supported along with single, double, and triple spacing.

If the report/file has neither ASA nor MCC controls, it is treated as basic single spacing.

Symbolic Substitution

Symbolic substitution is performed in variable operations when RuleSet statements are processed, when cover page files are loaded for processing, and when the console audit message is generated.

When RuleSet or cover page statements are processed they are scanned for symbols that can be replaced by values from the report. Symbols are marked by a leading “&” and a terminating “;”.

In batch mode

When running as a batch job, the symbolic values will be related to the current job, not the job that created the report.

&JobName;	Jobname of current job
&JobID;	JobID assigned to job
&ProcName;	Procname of current job
&StepName;	Step name of current step
&FormName;	Constant “FormName”
&DestID;	Constant “DestID”
&DSName;	Dataset name from REPORT=
&FlashName;	Constant “FLSH”
&FCBName;	Constant “FCB”
&LineCount;	0
&PageCount;	0
&Tag1;	See note below
&OurDomainName;	The domain name from the OurDomainName configuration statement
&UserID;	UserID that created the report
&Writer;	Constant “WRITER”
&DDName;	Constant “DDNAME”
&OutClass;	Constant “A”
&RuleSet;	Filename of current RuleSet
&RulesetRepID;	RuleSet Report ID (Note 1)

&POHost;	Name of current/active post office
&LDAPHost;	Name of LDAP server
&POIpAddress;	IP address of post office
&YY;	Current Year "01"
&YYYY;	Current Year "2001"
&Mon;	Current Month "10"
&DD;	Current Day "30"
&HH;	Current Hour "17"
&MM;	Current Minute "06"
&SS;	Current Second "59"

If a symbol is not recognized, it will be 'passed through' without changes.

Note1: Some variables, such as the RuleSetRepID, are not valid when processing the RuleSet statements; however, since the same substitution is performed when loading/processing a Script file, they may be appropriate.

Sample RuleSet

Here is an example of a Report RuleSet that would use the DestID value of the report to select an address list for routing and the report's FormName to select an Overlay definition:

```
Name: Stock form &FormName; Job(&Jobname;)  
Recognize by OutClass: F  
To: @LIST=ABC. ADRLIST. PDS(&DestID; )  
From: JES2Mail @&OurDomainName;  
PackageType: pdf  
Base64: yes  
FontSize: 9  
font: Courier  
Overlay: &FormName;
```

The **&Tag1;** symbol is only used when Symbolic Substitution is to create the Audit Message. The **&Tag1;** will contain either "Sent ok" or an error message related to the sending operation.

The **&PageCount;** symbol contains the 'page count' according to JES. Not all SYSOUTs in JES contain page count values.

Report RuleSet Statements

Each Report Description set should begin with an **ID:** statement, except the first that is the default, and should be terminated by a blank line.

/INSERT instruction

The **/INSERT** instruction can be used insert a file into the RuleSet.

The **/INSERT** must begin in the first column of the statement, and is followed by a filename. If the filename is less than nine characters in length, it is assumed to be a member of the CONFIG PDF. If the filename begins with "DD:", then it is a reference to

a JCL DD statement; otherwise the filename must be a fully qualified MVS dataset name.

`/INSERT` instructions can be nested 10 deep; that is, an “inserted” file can contain `/INSERT` instructions.

Action:

The **Action** statement is only needed when the action of **send** is not desired.

```

Action: Ignore
Action: Send
Action: Fail
Action: Mail (JES2Mail authorized mode)
Action: FTP (JES2FTP Authorized mode)
Action: FILE (JES2FTP Authorized mode)
Action: JES (JES2FTP Authorized Mode)

```

An action of **FAIL** will place the SYSOUT on hold, just as if there were some other kind of error in processing the report.

The action of **SEND** is the default if no action is specified. If the program is running in either JES2Mail or JES2FTP mode only, then the action of **SEND** is non-ambiguous. In JES2Mail mode **SEND** becomes **MAIL** and in JES2FTP mode **SEND** becomes **FTP**. However, if both modes are authorized, then the default **SEND** action can be set in the “DefaultSendAction” configuration statement. And finally, if both modes are authorized and not set in the config file, then the default is **MAIL**.

The action **MAIL** specifies an e-mail delivery. This is for JES2Mail mode.

The actions **FTP**, **FILE**, and **JES** specify actions for JES2FTP mode.

The action **Ignore** will ignore the report, it will be deleted from the JES queue as if it had been processed successfully.

AttachAsName:

When a file is delivered as an attachment, the filename is created by the Report Name and the Package Type. To override the filename created, the attachment’s filename can be explicitly supplied with this statement. For example:

```
AttachAsName: MYREP.PDF
```

If the suffix is `.zip` and the configuration file specifies `ZIPService=NATIVE`, then the file will be compressed and sent as a ZIP compliant file. For example:

```
AttachAsName: MyArchive.zip
```

See chapter on ZIP support for more information.

BCC:

Supply a Blind Carbon Copy address for the message. See “E-mail Addresses” below for a description of the parameter. There can be multiple **BCC:** statements for each report

set. Each **BCC:** can have multiple e-mail addresses, separated by a semicolon (;) or comma (,) For example:

```
BCC: GTamas@company.com
BCC: @LIST=SYSC.LIST.PDS(LIST1)
BCC: @LIST=LIST2
BCC: user1@comany1.com; user2@comany2.com
```

CC:

Supply a carbon copy address for the message. See “*E-mail Addresses*” below for a description of the parameters. There can be multiple **cc:** statements for each report set. Each **cc:** can have multiple e-mail addresses, separated by a semicolon (;) or comma (,). For example:

```
CC: GTamas@company.com
CC: @LIST=SYSC.LIST.PDS(LIST1)
CC: @LIST=LIST2
```

CoverPage:

When the report/file is sent as an attachment (as in PDF or Text formats), there needs to be a ‘cover sheet’ message. The default ‘cover sheet’ is a simple one-line message generated from the Report Name statement. If a different ‘**CoverPage**’ file is desired, this statement can point to the sequential file that contains it.

If no cover page is desired, specify “No”. Otherwise a default cover page is automatically generated. For example:

```
CoverPage: SYSC.COVER.PDS(COVER1)
CoverPage: Cover2
CoverPage: No
```

If the cover page is to be formatted with HTML, then the first character of the first line of the file *must* be a “<” (less-than sign). This is the trigger that JES2MAIL looks for to send the appropriate MIME header.

If the cover page is used in a non-report splitter operation, then the variable substitution is limited to the same values as the RuleSet statements. (Splitter Script cover pages have lots more choices).

Refer to the section on Symbolic Substitution for a list of all the available variables.

DOCInfoAuthor:

Specify the information to be included in the PDF or RTF Document Information fields. The default comes from the configuration file. Example:

```
DOCInfoAuthor: Sam Snead
```

DOCInfoCreator:

Specify the information to be included in the PDF or RTF Document Information fields. The default comes from the configuration file. Example:

DOCInfoCreator: JES2Mail on OS/390 Host

DOCInfoKeywords:

Specify the information to be included in the PDF or RTF Document Information fields. The default comes from the configuration file. Example:

DOCInfoKeywords: Invoice

DOCInfoProducer:

Specify the information to be included in the PDF or RTF Document Information fields. The default comes from the configuration file. Example:

DOCInfoProducer: JES2Mail

DOCInfoSubject:

Specify the information to be included in the PDF or RTF Document Information fields. The default comes from the configuration file. Example:

DOCInfoSubject: Monthly Invoices

DOCInfoTitle:

Specify the information to be included in the PDF or RTF Document Information fields. The default comes from the configuration file. Example:

DOCInfoTitle: Invoices

Font:

Select the desired output font name. If the PackageType is PDF, then it must match one of the supported base PDF fonts. If the PackageType is HTML, then the font name is passed through as given.

The valid **PDF** fonts are the following:

Courier	Helvetica-Bold	Times-Roman
Courier-Bold	Helvetica-Oblique	Times-Bold
Courier-Oblique	Helvetica-BoldOblique	Times-Italic
Courier-BoldOblique	Symbol	Times-BoldItalic
Helvetica		

For example:

Font: Courier

The valid **RTF** fonts consist of a font family followed by a specific font character set. The case is sensitive.

modern Courier New	(Default)
modern Pica	
roman Times New Roman	
roman Palatino	
swiss Arial	
script Course	

```
décor Old English
```

For example:

```
Font: swiss Arial
```

Note: with RTF if the target reader does not support the requested font, it will pick something close to what is available.

The **Font** statement should be placed after the **PackageType** statement.

FontSize:

The **FontSize** is very different for PDF and RTF vs. HTML format. The font size in PDF and RTF is in standard point size. For example, the text in this paragraph is **FontSize** of 10. A **FontSize** of 9 will enable a 132 column wide report to fit on a 11" wide page.

The font size for HTML is much 'looser'. The values can be from 1 to 7 with 1 being the smallest, 3 the default, and 7 the largest. If you specify a **FontSize** for an HTML **PackageType**, then the **Font name** must also be specified.

For example:

```
PackageType: PDF
FontSize: 9
PackageType: HTML
FontSize: 2
```

The **FontSize** can also be specified with a predefined symbolic value of **Tiny**, **Small**, **Medium**, **Large**, or **Huge**. Refer to the **FontSizeHTML** configuration statement for more information. The predefined values are:

Name	HTML	PDF	RTF
Tiny	1	4	4
Small	2	7	7
Medium	3	9	9
Large	4	12	12
Huge	5	16	16

For example:

```
PackageType: PDF
FontSize: Medium

PackageType: RTF
FontSize: tiny
```

From:

Supply a **From:** address for the message. If the "**From:**" is omitted, the message will be sent with a *null* return address. This is legitimate, but the recipient will not be able to reply to the message. The format of the address is the same as the e-mail address described below, except a *list* is not allowed. For example:

```
From: Sales Department sales@corp.com
```

ID: (Required)

This is the only required parameter for a Report Description set. The ID is used in various log messages generated for tracking purposes. The ID should be short, but the only limitation is the length of the line. For example:

```
ID: SALES1
```

LinesPerInch:

Normally the spacing of the text on the page is based on the font size. However, you can force the vertical line spacing by using the `LinesPerInch` statement. This does not affect the font size, but it does affect the calculation to determine the maximum lines per page when reading the report. For example:

```
LinesPerInch: 8
```

MaxLinesPerPage:

Normally, the 'working' maximum lines in a page are determined by the page size, margins, font size, and maybe the lines per inch. This figure can be overridden with this statement. This can be helpful when the input report is going to be 'printed' on an overlay that has yet to be specified by the script language. For example:

```
MaxLinesPerPage: 68
```

Name:

The "**Name:**" of the report is used to generate the 'cover page' for the attached report/file. For example:

```
Name: Daily Sales Report
```

Outline:

The "**Outline**" statement is used to specify that a PDF outline (or index) is to be generated from the contents of the report. A PDF outline is discussed more in the Report Script Language section of this manual. However, without using a script a simple "**Outline**" can be created with a single statement.

The **Outline** statement has two parameters: the first is a position in the report page, and the second is the length of the data.

The position is specified as `LineNo:ColumnNumber`. For example:

```
Outline 5:25 15
```

Will create an outline (index) from data starting in line 5, column 25, for a length of 15 characters.

An entry into the index will be made when the contents of the data field changes. For example, if a report has a number of pages, the outline will contain entries that point at the beginning of each group.

For multi-level outlines or where the outline data does not come from a fixed position on the page, the Report Script language can be used.

Overlay:

The “**Overlay**” has meaning if the **PackageType** is **PDF**. The **filename** specified contains **Overlay** definition statements. For example:

```
OverLay: Greenbar  
OverLay: AAA.JES2MAIL.CNTL(INVOICE)
```

PackageType:

The “**PackageType**” determines how the report/file is to be formatted, encoded, and attached to the e-mail message.

The choices for **PackageType** are as follows:

TEXT	Attach the report/file as a simple text file. Convert the Carriage Control to CR/LF and FF characters.
TEXTASA	Attach the report/file as a simple text file. Convert (or pass through) the Carriage Control to standard ASA control codes in the first position of each line.
PDF	Attach the report/file as an Adobe PDF file. Use the carriage control to format the text into the PDF ‘pages’.
HTML	Send the report/file as the ‘body’ of the e-mail message. Any special HTML characters will be encapsulated so that they don’t interfere with HTML processing.
HTMLASIS	Send the report/file as the ‘body’ of the e-mail message. This value will trigger the program to ‘set up’ the proper MIME header lines so that the e-mail client will know that the message body is HTML. However, no attempt to ‘sanitize’ the content is performed.
RTF	Send the report/file as an attached RTF formatted file.
DATA	Process file as non-report data. See section “Processing non-report data”.

For example:

```
PackageType: PDF  
PackageType: TEXT
```

PageHeight:

This parameter has meaning for PDF formatting. “**PageHeight**” sets the PDF page height in inches. For example:

```
PageHeight: 11
PageHeight: 8.5
```

PageSize:

An alternative to specifying the PDF page with page width and height is to use the **PageSize** statement. If you use the **PageSize**, you should use the **Orientation** statement as well. Here are the page sizes and their values; the default is Letter.

PageSize	Inches	Inches
Letter	8.5	11
Legal	8.5	14
A2	16.54	23.39
A3	11.69	16.54
A4	8.27	11.69
A5	5.83	8.27
A3X	12.69	17.53
A4X	9.26	12.69

For example:

```
PageSize: A4
Orientation: L
```

PageWidth:

This parameter has meaning for PDF formatting. “**PageWidth**” sets the PDF page width in inches. For example:

```
PageWidth: 11
PageWidth: 8.5
```

PDFCenterWindow:

To request that *Acrobat* open the document’s window in the center of the screen, use the following statement:

```
PDFCenterWindow: Yes
```

The default is “NO”.

ProtectLevel:

If the PDF document is to be encrypted, the level of encryption is specified with this statement. The choices are:

```
off          No encryption
```

Standard Standard level of encryptions is 40 bit

Full Full encryption is 128 bit

For example:

```
ProtectLevel: Standard
```

ProtectOp:

When the ProtectLevel specifies encryption, then various client (Acrobat) functions can be restricted. Several options can be specified, and some are only available with Full (128 bit) encryption. The choices are:

NoPrint Don't allow the client to print the document.

NoModify Don't allow modifications. This restricts Acrobat's 'text touch-up' tool from changing contents.

NoCopy Don't allow a Copy operation that would copy text to the PC notepad.

NoAnnots Don't allow any Annotations, or acroform building.

NoAcroform Don't allow Acroform usage. *

NoExtract Don't allow extraction of text and graphics (in support of accessibility to disabled users or for other purposes). *

NoAssemble Don't allow assembly of document (insert, rotate, or delete pages, nor create bookmarks or thumbnail images). *

NoImagePrint Don't allow printing to 'high quality' printer. Printing is limited to a low level representation, possibility of degraded quality. *

* These functions require the protect level to be FULL 128 bit encryption.

The restriction is limited by the support of the client product, such as the Acrobat reader. It is possible that some other PDF 'reader' program could not abide by the PDF standards and allow such operations.

Parameters can be entered on one statement or with several, for example:

```
ProtectOp: NoPrint NoModify NoCopy  
ProtectOp: NoAssemble  
ProtectOp: NoAnnots
```

ProtectOwnerPassword:

Assign the PDF document an 'owner' password. If the ProtectLevel specifies encryption and no owner password is supplied, then a random 'owner' password is

generated. The protection level and allowable operations for a PDF document cannot be altered by the Acrobat client, without entering the proper owner password.

The password can be up to 32 characters in length, and the recipient must enter the same password precisely, case counts. For example:

```
ProtectOwnerPassword: George
ProtectOwnerPassword: (George)
```

ProtectUserPassword:

To assign a ‘user’ password to the document, use this statement. If the `ProtectLevel` specifies encryption, but no user password is specified, then the recipient will be able to view the document without any password prompt. Their options may be restricted by the `Protectop` parameters. As in the `ProtectOwnerPassword`, if the password is enclosed in parentheses, then it is looked up in the table specified by `ProtectPasswordTable`.

For example:

```
ProtectUserPassword: sam
```

Sanitize:

The `sanitize` option instructs JES2MAIL to scrub the report of any non-displayable characters before transmitting them as part of the report.

There are four levels of sanitize:

- No:** No scrubbing is performed.
- Nulls** Null characters (x'00') are removed.
- CNTL:** Characters from x'00' to x'3f' are removed.
- YES or FULL:** All characters that do not pass the C/C++ “isgraph” function are replaced with spaces.

This is an option since it takes additional computing power to scan a report for non-displayable characters and changing them to spaces. The default is determined by the `sanitizeReports` configuration statement’s value. If that is not specified, then the default is “No”.

For example:

```
Sani ti ze: Yes
Sani ti ze: Nulls
Sani ti ze: Cntl
```

SuppressCC:

By default a form-feed character is inserted at the beginning of each report page formatted as a text file. This can be suppressed by using a command such as:

```
SuppressCC: Yes
```

To:

Supply a primary address for the message. See “*E-mail Addresses*” below for a description of the parameters. There can be multiple **TO:** statements for each report set. Each **TO:** can have multiple e-mail addresses, separated by a semicolon (;) or comma(,).

Use the keyword “**DEFAULT**” when the destination is to be copied from the default (first) report processing rule. For example:

```
To: GTamas@company.com  
To: @LIST=SYSC.LIST.PDS(LIST1)  
To: Sally Smith <SallyS@company.com>  
To: user1@company1.com; user2@company2.com  
To: Default
```

ZIPMemberName

When NATIVE ZIP support is enabled in the configuration file, and the AttachAsFilename specifies a .zip file suffix, then ZIPMemberName statement can be used to specify the filename within the ZIP archive for the report. If no ZIPMemberName is specified, then the inside report filename will be the same as the ZIP archive name, with the .zip suffix changed to match the package type (.txt, .pdf etc.).

For example:

```
ZIPMemberName: Report.txt
```

When the Report file specifies .zip, then a ZIP archive is opened, and not only the main report file, but a subsequent extract data file (comma delimited) will also be inserted into the ZIP archive.

See the chapter on ZIP support for more information.

E-mail Addressing

E-mail addresses are used in the **From:**, **To:**, **CC:** and **BCC:** statements. An e-mail address can be as simple as `userid@domainName`. For example:

```
To: user@domain.com
```

If a ‘human readable’ name is available it can be included in the e-mail address, but then the e-mail address must be enclosed in `< >`. For example:

```
To: Bob LaBayne <bobl2casi@soft.com>
```

Multiple e-mail addresses can be supplied by stringing together the addresses and separating with either a semicolon (;) or comma (.). For example:

```
To: user@domain.com; bend@casisoft.com  
To: Sally Smith <Ssmith@domain>; User@domain.com
```

Route Lists

If the desire is to send the message to a large number of recipients, it is best to use a route list. A route list is simply a sequential file with e-mail addresses. Whereas the e-mail address supplied in the **To:** and **CC:** will appear in the message body, when a list is used, the message body will only contain the list name. Thus if the list contains 100 recipients, only one line will appear in the e-mail message header lines.

To designate a route list, in any of the **To:**, **CC:** or **BCC:** statements simply enter "@LIST=" followed by the OS/390 dataset name. For example:

```
To: ListName <@LIST=SYSC.LIST.PDS(LIST1)>
```

Each 'record' of the list file should contain a single e-mail address. If the first character of a route list entry begins with a semicolon (;) or asterisk (*), it is treated as a comment and bypassed.

Here is an example of the above mentioned list file:

```
user@domain.com  
Randy Benderman <bend@casisoft.com>  
Bob LaBayne <bob12@casisoft.com>  
Pam@data21.com
```

Note: Normally, the final message body is presented with the full distribution text, "<@LIST=xxx>", but we have found that some mail servers are not happy with this syntax. You may need to use the configuration option, **ReplaceAtListText** to have the offending text changed to something acceptable to your mail server.

7

ZIP Compression Support

JES2Mail has several levels of compression support PKWare's PKZIP for MVS, Data21's ZIP/390 and the build-in Native compression described here.

Configuration File

In the configuration file, specify NATIVE in the ZIPService parameter, as in:

```
ZipService: NATIVE
```

Specify the maximum for the archive file. This value is used to allocate storage to hold the archive as it is being built. Since most email administrators put a limit on the size of attachments, setting a value in the 2meg to 4meg range is probably all that should be specified. (Basically, if you are sending larger files than that, you probably should be considering using JES2FTP to put the archive on an FTP server, and sending an email notice with a link.) For example:

```
ZipArchiveSize: 4000000
```

Specify the maximum size of the 'original' data file. This value is used to allocate storage for the uncompressed file prior to being compressed and added to the ZIP archive storage area. For example:

```
ZipWorkBufferSize: 2000000
```

Ruleset

AttachAsName

When processing reports, the 'trigger' to invoke ZIP compression is to specify a destination filename with a .zip suffix. This is done with the AttachAsName statement. For example:

```
AttachAsName: hlq.archive.zip
```

This statement defines the ZIP archive filename, instead of the report's filename.

ZIPMemberName

A ZIPMemberName statement can be used to specify the filename 'within' the ZIP archive for the report. If no ZIPMemberName is specified, then the inside report filename will be the same as the ZIP archive name, with the .zip suffix changed to match the package type (.txt, .pdf, etc.).

For example:

```
ZIPMemberName: Report.txt
```

When the AttachAsName parameter specifies .zip, then a ZIP archive is opened, and not only the main report file whose name is specified with the ZipMemberName parameter, but any subsequent extract data file (comma delimited) will also be inserted into the ZIP archive.

ZIP compression doesn't compress PDF files much, since they are already mostly compressed files. Where a PDF gets about 80% reduction normally, ZIPping a PDF might bring that up to 85%. However, for text reports or extract data files, you can see compression in the 75 to 90% range depending on the contents.

ATTACH statement

The ATTACH statement can be used in two situations, in a simple message and in a)MessageHeader section of a RPL script.

Normally, no matter how many **ATTACH** statements are specified, all of the files are ZIPped into a single ZIP file, called a ZIP Archive. However the ATTACH statement with the "ZipClose" option can be added and then any subsequent ATTACH statements will begin a new archive file.

The recipient can unzip the archive files with various tools.

When the ZIP mechanism supports encryption, then each individual file within the ZIP archive file can have a password.

The storage for the ZIP process is acquired when the first **ATTACH** statement with a ZIP request is processed, and it is all released when all of the attachments have been zipped and shipped.

If the ZIP option is specified, and no ZIP services are available, the file is sent without compression

The **ATTACH**: statement is followed by the filename and various options.

```
TYPE: file-type
AS: new-filename
NOTRIM
BASE64
ZIP
ZipName: archive-name
```

Source-name The filename to be attached immediately follows the ATTACH statement. The filename must be a fully qualified filename, or a DD: reference. If the filename specifies a PDS(E), the member name can be a wildcard of asterisk (*) for multiple characters or question mark (?) for a single character. When a wildcard is detected, it is expanded and treated as an individual ATTACH statement for each qualifying filename.

File-Type. If the filetype is "text", "txt", "html", or "htm", then the file is treated as TEXT, and will be translated from EBCDIC to ASCII. Any other value will force the file to be treated as BINARY, and will be attached as is, no translation.

New-filename can specify a 'PC friendly' filename for the attachment. When this is not specified, then the source filename is used. If the source filename specifies a PDS(E) member, you can use a special symbolic, &member; to reference the member name.

For example:

```
Attach: hlq.file.pds(Report1) as: &Member;.txt
```

NOTRIM. Normally, when TEXT files are processed, trailing spaces are removed from the record before the ASCII new-line character is appended to each line. If you want to suppress the trimming, add the **NOTRIM** option to the statement.

ZIP. If zip service is available, the file is ZIP compressed into a .ZIP archive file.

ZipName: name The default ZIP archive name, that is the name that the recipient will see as the attachment's filename, is "Archive.ZIP". To specify an archive name, include this parameter followed by the new name on the first ATTACH statement for the archive.

ATTACH statements can be continued by placing a dash at the end of the line, and continuing on the next line.

Examples:

```
Attach: USER.FILE.SOURCE(TEXT1) Type: text
Attach: USER.OBJECT.PDS(OBJ1) Type: binary
```

```

Attach: ABC.FILE Type: Binary As File.bin
Attach: ABC.Longfilename.souce(text) -
Type: Text As: "Long PC filename.txt"
Attach: user.filename Type: text -
as:'PCFile.txt' zip
Attach: hlq.filename.pds(j2x*) type: text as:&membername;.jcl -
Zip

```

Native Compression Support

Since NATIVE compression is built-in, all that is needed is the configuration statement that specifies ZIPSERVICE=NATIVE. Here is an example of a job that ZIPs and ships:

```

//JES2MAIL EXEC PGM=J2MOE
//LOGFILE DD SYSOUT=*
//CONFIG DD DSN=USER.JES2MAIL.CONFIG,DISP=SHR
//STEPLIB DD DSN=USER.JES2MAIL.LOAD,DISP=SHR
//SYSIN DD *
FILE=DD: MESSAGE
END
//MESSAGE DD *
To: user@domain.com
Date:
Subject: Test message zip
Attach: ABC.FILE.ONE type: text -
as:'abc.txt'
Attach: USER.file.two type: text -
as:'user2.txt' zip
Attach: user.binary.data(binfile1) type: bin -
as:'binfile.dat' zip

This is a test message. with native zipping.
This is the job: &ZJobName; -&ZJobno;
.

```

In this example, the final delivered e-mail will contain two attachments. The first will be the "ABC.TXT" attachment for which no zipping was specified. The second attachment will be an archive (zip) file named "USER2.ZIP" whose name is derived from the first "as" filename specified with a ZIP parameter. This archive file will contain both the "USER2.TXT" file and the "BINFILE.DAT" file. If a different archive file name was desired this could have been specified by adding the ZIPNAME parameter to the Attach statement such as this:

```

Attach: USER.file.two type: text -
as:'user2.txt' Zip Zipname: myarchive.zip

```

Examples:

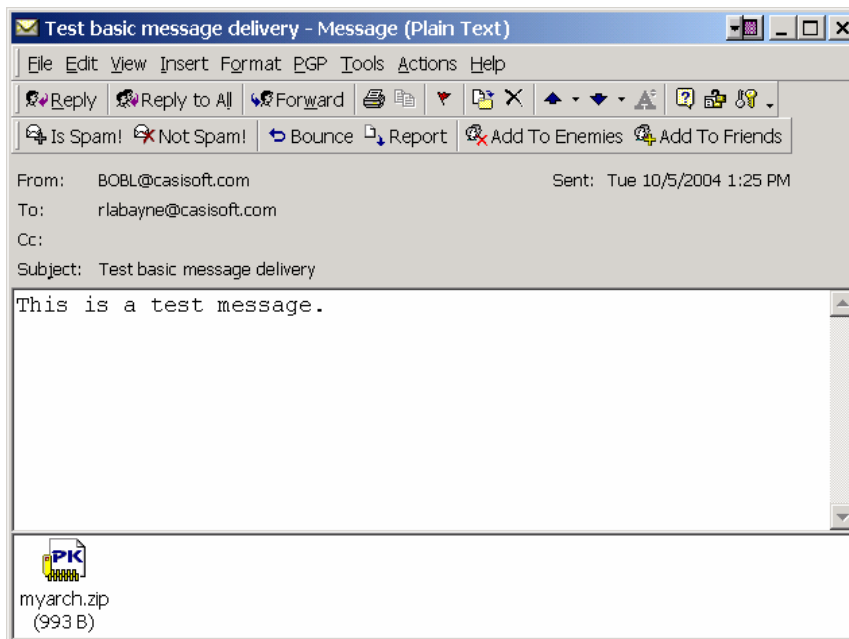
Example of JES2Mail ATTACHments

Here is an example of a simple non-report email message.

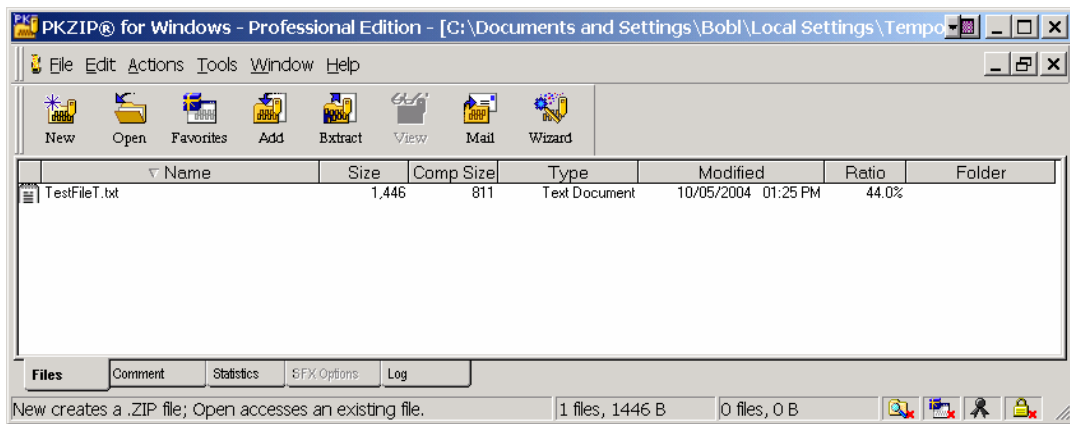
```
//JES2MAIL EXEC PGM=J2X0E
//LOGFILE DD SYSOUT=*
//CONFIG DD DSN=hlq.JES2MAIL.CONFIG,DISP=SHR
//STEPLIB DD DSN=hlq.JES2MAIL.LOAD,DISP=SHR
//SYSIN DD *
TRACE: Full
Message=DD: MESSAGE
END
//MESSAGE DD *
To: rlabayne@casisoft.com
From: &zuserid;@casisoft.com
Date:
Subject: Test basic message delivery
Attach: hlq.jes2mail.proclib(j2xm52a) type: text as:File1.txt zip -
        zipname myarch.zip
```

This is a test message.

Here is what the Outlook email client shows for the message:



Opening the “myarch.zip” file with PKZIP you get:



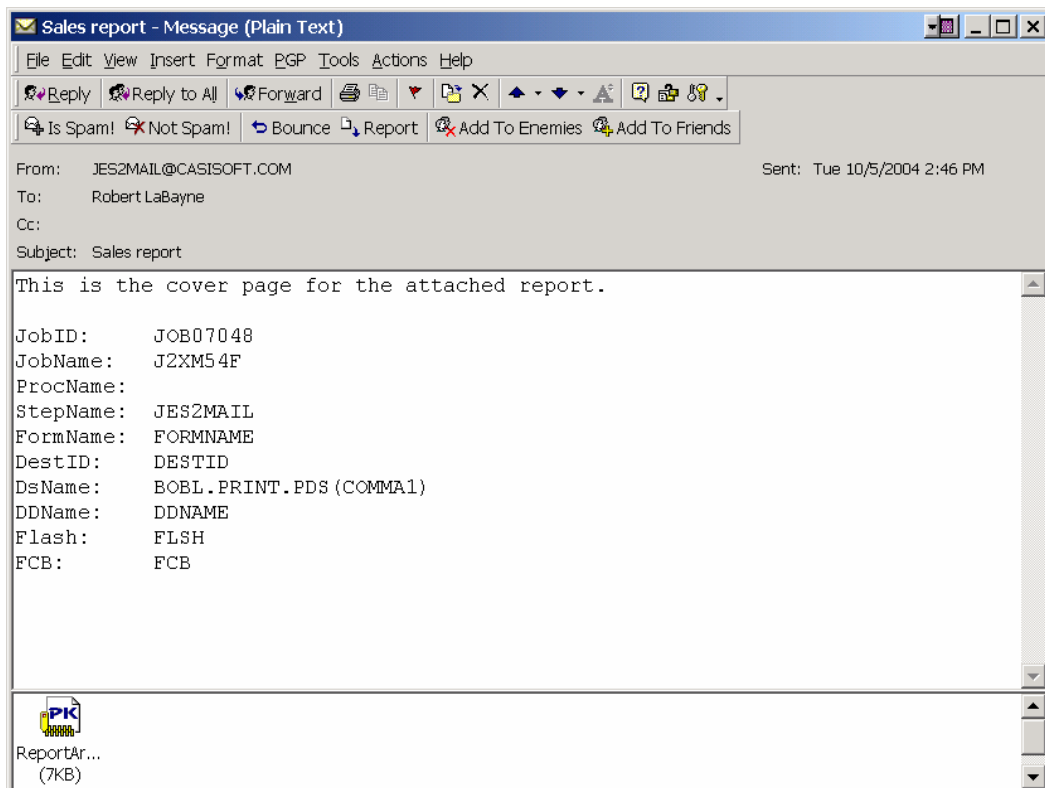
You can see that the “ZIPName myarch.zip” specifies the filename for the .zip file, and inside the archive, the “as:TestFile.txt” becomes the member name of the file.

Example of JES2Mail Main Report

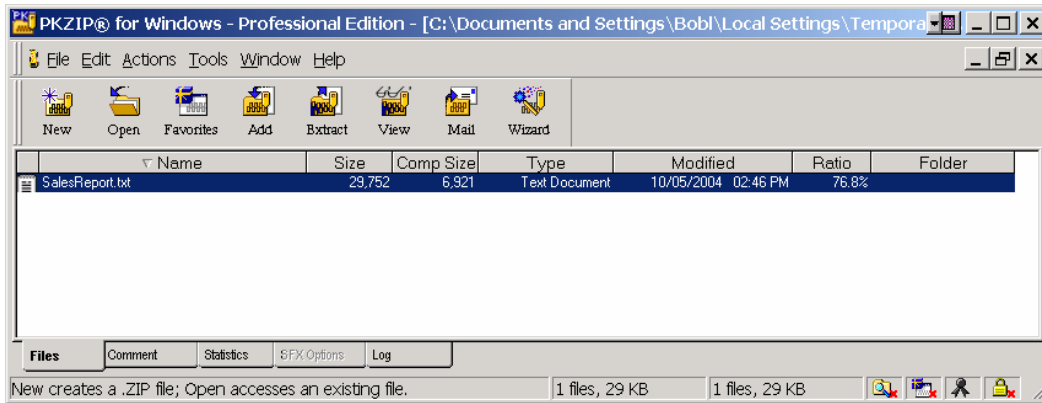
Here is a basic report being sent in TEXT format, but enclosed in a ZIP archive.

```
//JES2MAIL EXEC PGM=J2X0E
//LOGFILE DD SYSOUT=*
//CONFIG DD DSN=hlq.JES2MAIL.CONFIG,DISP=SHR
//STEPLIB DD DSN=hlq.JES2MAIL.LOAD,DISP=SHR
//SYSIN DD *
TRACE=full
RULESET=DD:RULE
REPORT=hlq.report.file
END
//RULE DD *
NAME: Sales report
To: user@domain.com
From: JES2MAIL@Domain.COM
PackageType: text
SuppressCC: Yes
AttachAsName: ReportArchive.zip
ZIPMembername: SalesReport.txt
Action: Mail
```

The email client shows:



Opening the PKZIP archive we see:



Opening the report file in notepad, we get:

The screenshot shows the Notepad window displaying the contents of SalesReport.txt. The text is a comma-separated sales report for the division of North, listing various companies and their sales figures for six different products (A through F).

COMMA SALES								
DIVISION: NORTH								
COMPANY NAME	ABCD	ABC	PRODUCT-A	PRODUCT-B	PRODUCT-C	PRODUCT-D	PRODUCT-E	PRODUCT-F
INTERSTATE STOCKYARDS			18,506	36281	58649	43323	28934	85997
WYTG BROADCASTING INC			60506	33567	67208	61974	51073	56008
HOMESTEAD ESTATES			70635	78414	97709	18095	76115	14549
WYTG BROADCASTING INC			55643		45303	13101	64485	3490
HEALTHY VINEGARS INC			65653	11628	97501	1230	15284	96356
SAFARI IMPORTS LTD			3511	69622	35148	98108	11670	14741
PURE GROWTH DOG FOOD INC			39184	45237		99552	30208	12030
RENTAL PROPERTIES INC			59019	68285	37694	56656	93957	35367
WYTG BROADCASTING INC			64321	89151	21679	67779	3791	73670
INTERSTATE STOCKYARDS			57264	15003	52298	47045	31461	
ACME SLEDGE HAMMERS INC			92525	42772	83157	94111	36132	79344
SAFARI IMPORTS LTD			18898	30203	35452	72239	45598	46696
YOUNG & BROWN INC			33244	90470	64843	35210	26720	98383
RENTAL PROPERTIES INC			23882	50187	62245	33716	85267	92744
INTERSTATE STOCKYARDS			96582	13077	70328	2304	1484	32346
INTERSTATE STOCKYARDS			93745	69167	34615	18960	32860	13730
SAFARI IMPORTS LTD			70524	25797	62445	22673	920	5202
INTERSTATE STOCKYARDS			22213	50639	37718	33690	14865	40998
HOMESTEAD ESTATES			96282	71345	55528	98269	67765	40134

Readers' Comments – We'd Like to Hear from You

If you especially like or dislike anything about this manual, please e-mail us at the address listed below. Please include your name, address and telephone number if you would like a reply.

Please include the product name, JES2FTP or JES2Mail, and version and release level you are running. These are always displayed in the first line of the LOGFILE trace.

Email Address: support@casisoft.com