



***PostalOne!*<sup>®</sup> Technical Guide**  
**Version 7.0.0**

**May 2004**

## PREFACE

This document is a comprehensive technical guide that outlines steps for exchanging electronic data with the *PostalOne!*® system. The intended audience is business mailers who create/use mailing data in the preparation and/or production of business mail and who also want to send that electronic mailing data to the *PostalOne!* system for the purpose of conducting business electronically in lieu of the hardcopy forms and processes that were traditionally used in business mail acceptance. Although this guide provides an overview of the *PostalOne!* system, the intended recipient of this document is the person or team that will make necessary technical infrastructure preparations/alterations and conduct testing to ensure electronic data is consistently and reliably sent to the *PostalOne!* system.

This guide is organized as follows: Chapters 1 and 2 provide an overview of the *PostalOne!* program benefits, the basic requirements for participation in the program, and how to apply for participation. Chapters 3 through 4 provide detailed technical information about the *PostalOne!* application. Chapter 5 provides information on who to contact for help. Appendices A, B and C contain technical information about the electronic data used by the system. Appendices D and E contain error messages that you may need to reference.

On behalf of the *PostalOne!* team, we would like to welcome you into the program. You will be joining many other business mailers who are successfully using the *PostalOne!* system to improve the collaboration with the U. S. Postal Service®. We welcome your feedback and look forward to working with you.

## TABLE OF CONTENTS

<b>1</b>	<b>PROGRAM OVERVIEW</b> .....	<b>1</b>
<b>2</b>	<b>APPLYING FOR PARTICIPATION</b> .....	<b>1</b>
<b>3</b>	<b>SENDING PROPERLY CONFIGURED MAILING FILES</b> .....	<b>1</b>
3.1	KEY TECHNICAL REQUIREMENTS .....	2
3.1.1	Internet Connection .....	2
3.1.2	Formatted Mailing Files .....	2
3.1.3	Methods of Transferring Files (Manual and Batch).....	2
3.2	TECHNICAL/OPERATIONAL PREREQUISITES .....	3
3.2.1	Hardware .....	3
3.2.2	Software .....	4
3.2.3	Network .....	4
3.2.4	Infrastructure .....	4
3.2.5	Information/Manuals .....	4
3.2.6	User License Code .....	5
3.2.7	Security Certificates .....	5
3.3	CONFIGURING MAILING FILES FOR PROCESSING.....	5
3.3.1	Required Files and Fields .....	5
3.3.2	Typical File Formatting Problems.....	5
3.3.3	Sending Job Updates .....	6
3.3.3.1	File/Record Level Status .....	8
3.3.3.2	Rules for Sending Updates .....	8
3.3.3.3	Commands.....	9
3.3.4	File Processing Overview.....	10
3.4	TRANSFERRING FILES .....	11
3.4.1	Validating Files .....	11
3.4.1.1	Validation Constraints .....	12
3.4.1.2	Relationship Constraints Description .....	15
3.4.1.3	Validation Log Errors.....	18
3.4.2	Performing the File Transfer .....	19
<b>4</b>	<b>BATCH PROCESSING</b> .....	<b>20</b>
4.1	PREREQUISITES FOR BATCH PROCESSING .....	21
4.1.1	Hardware .....	21
4.1.2	Software .....	21
4.1.3	Network .....	21
4.1.4	Downloading and Configuring the NT Batch Processor .....	22
4.1.5	Downloading and Configuring the UNIX Batch Processor.....	24
4.1.6	Digital Certificates/Security .....	27
4.1.6.1	Creating a Keystore .....	27
4.1.6.2	Creating Certificate Signing Requests .....	28
4.1.6.3	Submitting Your Certificate Signing Request.....	28
4.1.6.4	Adding a Signed Certificate to a Keystore .....	28
4.1.6.5	Enabling Encryption on the Batch Processor Client .....	29
4.1.6.6	Testing Your Signed Certificate .....	29
4.1.6.7	Testing Batch Processor Setup .....	30
4.1.7	Scheduling Batch Jobs (Running the Batch Processor).....	30
4.2	FILE TRANSFER ERROR CODES AND MESSAGES.....	30
<b>5</b>	<b>CUSTOMER SUPPORT</b> .....	<b>30</b>
	<b>APPENDIX A. MAIL.DAT FILE DEFINITIONS</b> .....	<b>31</b>

<b>APPENDIX B. STATUS RECEIPT FILE LAYOUT .....</b>	<b>33</b>
B-1 STATUS RECEIPT FILE CONTENT .....	33
B-2 RECORD TYPES AND RECORD FORMAT.....	33
<b>APPENDIX C. POSTAGE STATEMENT RECEIPT FILE LAYOUT.....</b>	<b>37</b>
C-1 POSTAGE STATEMENT RECEIPT FILE CONTENT .....	37
Summary Record.....	37
Receipt Header Information (Summary Record) .....	37
Job Information (Summary Record).....	38
Address Information (Summary Record) .....	38
Mailing information (Summary Record) .....	39
Computation Record.....	41
Receipt Header Information (Computation Record) .....	42
Job Information (Computation Record).....	42
Postage Computation Information (Detailed Calculations) .....	42
C-2 RECORD TYPES AND RECORD FORMAT .....	42
<b>APPENDIX D. VALIDATION LOG FILE ERROR MESSAGES.....</b>	<b>52</b>
<b>APPENDIX E. FILE TRANSFER ERROR MESSAGES .....</b>	<b>56</b>
<b>INDEX .....</b>	<b>65</b>

#### LIST OF FIGURES

Figure 3-1. The <i>PostalOne!</i> System—Electronic collaboration via the Internet.....	2
Figure 3-2. Sending Job Updates to the <i>PostalOne!</i> System.....	8
Figure 3-3. Mail.dat File Transfer Process.....	11

#### LIST OF TABLES

Table 3-1. RAM Requirements Based on Net Job Size .....	3
Table 3-2. Files Included in Original and Update Transfers .....	7
Table 3-3. File and Record Level Status Values .....	8
Table 3-4. Allowable Container Status Values .....	9
Table 3-5. RAM Requirements Based on Net Job Size .....	12
Table 3-6. Presentation Category Values Required to Pass Validation .....	13
Table 4-1. RAM Requirements Based on Net Job Size .....	21
Table 4-2. Postal1.ini Parameters for NT .....	23
Table 4-3. Postal1.ini Parameters for UNIX.....	25
Table A-1. Mail.dat Files and Definitions .....	31
Table D-1. Validation Log File Error Messages.....	52
Table E-1. File Transfer Error Messages.....	56

# 1 Program Overview

The *PostalOne!*® system is an electronic suite of services being developed for business mailers by the United States Postal Service® to dramatically improve the mailing process. The system takes advantage of the advances in technology and leverages them to significantly improve and simplify the mailing experience.

The system enables customers to electronically send detailed information using the mailing industry's standardized Mail.dat® format. Customers using the system no longer have to complete manual postage statements because they can be automatically generated by the *PostalOne!* system. In fact, much of the acceptance and verification process has been automated, enabling a faster and more accurate method of accepting, assessing, and finalizing postage statements. The electronic documentation provided by the system means you no longer have to physically bring hardcopy presort documentation to the acceptance unit or retain it for storage.

Customers can download an estimated postage summary, view individual postage statements within a mailing, see a running total of postage statement activity, and view the reconciliation of the information at the end of a mailing information exchange process (mailing job). By automating and incorporating payment processing into the *PostalOne!* system, payments for mailing jobs can be electronically processed.

As an added benefit, the Transportation Management function of the *PostalOne!* system provides select users with the ability to make surface and air transportation assignments and route mail in the mailer's own plant. This capability is a great benefit to many of the larger First-Class Mail® customers because they can track mailings more accurately by knowing the detailed routing.

## 2 Applying for Participation

Applying for and obtaining access to the *PostalOne!* system is quick and easy. We have made the application process available online through our website at <http://www.usps.com/postalone>. On the right, near the top of the page under New Customers, click the **Learn More** button. Then click the **Apply Online** link at the bottom of the Business Customer section. There, you can complete your online application. Following completion, our Customer Care Center reviews the application to ensure that we have accurately established your account. Within 24-48 hours of completing your online application, you will receive the *PostalOne!* Welcome Kit. The kit contains additional information about the many features of the *PostalOne!* system as well as the program's Data Exchange Agreement.

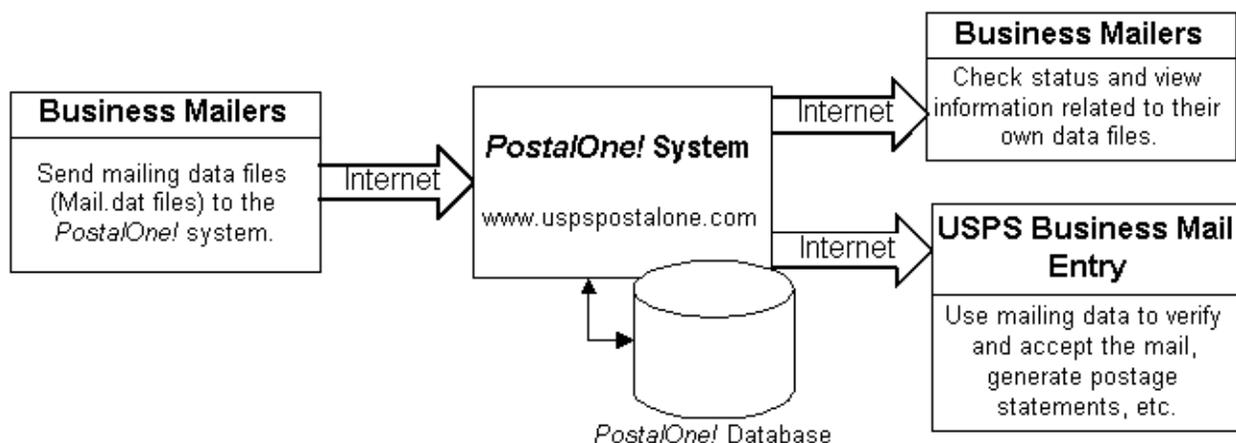
**NOTE:** All companies participating in the *PostalOne!* program are required to sign an agreement, which details the participant's rights and responsibilities in this electronic arena.

## 3 Sending Properly Configured Mailing Files

To successfully conduct business electronically in the *PostalOne!* system, you will send mailing files via the Internet. These mailing files must be in the mailing industry format called Mail.dat. Without this exchange of information, electronic collaboration is not possible.

Mail.dat files are processed and stored in the *PostalOne!* system. Postal Service personnel access these files over the Internet to conduct business. For example, Postal Service acceptance clerks use these detailed mailing data files in the mail verification and acceptance process. The *PostalOne!* system uses the mailing files to generate postage statements for customers who have presented mailings to the Postal Service.

The figure below illustrates the relationship between the system and its users. It shows how business mailers submit mailing data to the *PostalOne!* system and how that data is retrieved by Postal Service business mail entry units.



**Figure 3-1. The *PostalOne!* System–Electronic collaboration via the Internet**

**NOTE:** Mail.dat is a registered trademark of International Digital Enterprise Alliance, Inc. (IDEAlliance©) 2001-2002. IDEAlliance (<http://www.idealliance.org/>) is a not-for-profit membership organization, whose mission is to advance user-driven, cross-industry solutions for all publishing and content-related processes by developing standards, fostering business alliances, and identifying best practices.

## 3.1 Key Technical Requirements

### 3.1.1 Internet Connection

The minimum required speed for connectivity is 56kps. For optimal performance, we suggest that users with file sizes greater than one megabyte (MB) use some form of high-speed Internet access, such as T1 line, DSL, cable modem, etc.

### 3.1.2 Formatted Mailing Files

To participate in the *PostalOne!* program, the files and fields you send must comply with *PostalOne!* implementation of the *Mail.dat File Specification*. Additional technical details concerning Mail.dat file format and content, as used by the *PostalOne!* system, are provided in Section 3.3, Configuring Mailing Files for Processing. As a participant, you must successfully transfer Mail.dat files to the *PostalOne!* system.

### 3.1.3 Methods of Transferring Files (Manual and Batch)

The *PostalOne!* system provides you with a choice of transfer methods for your mailing data files (Mail.dat files). You can transfer mailing data files manually (Manual mode) or unattended (Batch mode).

1. **Manual mode.** In Manual mode, customers log on, select the files they want to send, and then upload them. This mode is called Manual because a person must actively use the *PostalOne!* system and be present while the file is transferred. The information presented in this chapter assumes you will make Manual mode file transfers to the system.
2. **Batch mode.** Batch mode allows customers to schedule file transfers to take place automatically, for example at 2 a.m. Once Batch mode is set up, no one needs to be present when the file transfer takes place. This capability requires additional infrastructure and configuration work. If you plan to send files in Batch mode, see Chapter 4, Batch Processing.

When using Batch mode, the *PostalOne!* servers return file transfer status feedback to the sender's workstation automatically. The feedback information (receipt file) is written to the client's workstation in either XML or character delimited ASCII text formats. You can load the receipt files into a spreadsheet or your own database for storage and viewing. For more information on the receipt file format and its contents, see Appendix B. Status Receipt File Layout.

## 3.2 Technical/Operational Prerequisites

As previously discussed, the system is Internet-based and the basis for conducting business electronically is the successful exchange of mailing data information between business mailers and the Postal Service. As a result, an Internet-based infrastructure must exist that is compatible with the *PostalOne!* system.

This section provides details on all of the technical and operational prerequisites necessary to successfully send mailing data files to the system. Topical areas include:

1. **Hardware prerequisites.** The base requirements for the hardware elements of the computer system that will access and send mailing data files to the *PostalOne!* system such as the recommended physical memory amounts. For more information on hardware prerequisites, see Section 3.2.1, Hardware.
2. **Software prerequisites.** The base requirements for the software elements of the computer system that will access and send mailing data files to the *PostalOne!* system, such as browser software versions. For more information on software prerequisites, see Section 3.2.2, Software.
3. **Networking prerequisites.** The base requirements for the networking elements of the computer system that will access and send mailing data files to the system, such as firewall settings. For more information on network prerequisites, see Section 3.2.3, Network.
4. **Mail.dat prerequisites.** The base requirements for the actual files sent. The *PostalOne!* system complies with the *Mail.dat File Specification*. As a result, mailing files must comply with this specification. We strongly recommend that you obtain and read the specification. For more information, see Section 3.2.5, Information/Manuals.
5. **User License Code (ULC) prerequisites.** To use Mail.dat, your company must have a ULC, also called a Provider Code. Business mailers using Mail.dat should already have a ULC; if not, they need to get one before continuing. For more information, see Section 3.2.6, User License Code.
6. **Optional digital certificate prerequisites.** To send mailing files in Batch mode, your company must obtain and install a digital security certificate. For digital certificate requirements, see Section 3.2.7, Security Certificates and Chapter 4, Batch Processing.

### 3.2.1 Hardware

The minimum workstation requirements to send data files are a PC with a Pentium II 266 MHz processor, Windows 95/98 operating system, and 64 MB RAM. Depending on the size of the Mail.dat job, RAM requirements vary. The table below details the RAM requirements:

**Table 3-1. RAM Requirements Based on Net Job Size**

Net Size of Mail.dat Job (MB) *	Recommended Physical RAM
0 – 10	64 MB
11 – 25	128 MB
26 – 50	256 MB

\* The net size of a Mail.dat job is calculated as the "total size of all files" associated with a job, minus the size of the PQT, SNR, and SPR files.

**NOTE:** UNIX workstations can also be used to send mailing data files in Batch processing mode. For more details, see Chapter 4, Batch Processing.

### 3.2.2 Software

Windows workstations used for transferring files should be configured using the following software:

- A text editor or third-party Mail.dat viewer. This software is necessary to analyze the contents of your Mail.dat files and resolve any file format or content problems reported by the *PostalOne!* system.
- Microsoft Internet Explorer 5.0 or later with the Java™ Virtual Machine (JVM) plug-in or Netscape® Communicator 6.0 or later with JVM plug-in. The JVM should be version 1.4.2 (see the next item).
- Because newer Microsoft operating systems (such as Windows® XP) no longer support some Java software, users with new installations of Microsoft operating systems must now install Sun Microsystems Java Virtual Machine 1.4.2 or greater on their browsers. If you do not do this, the *PostalOne!* functions for file validation and manual file transfer will not work. The Batch Processor has a different requirement.

To install the JVM, ensure you have administrator rights on your computer, then click the links on the Sun Microsystems site at: <http://www.java.com/>. If a security warning dialog box opens, click **Yes** to install, and follow any directions. If you do not have administrator rights, contact your system administrator. Once installed, the JVM automatically runs whenever necessary.

For information about batch mode requirements, see Chapter 4, Batch Processing.

### 3.2.3 Network

The File Transfer software (Manual mode) uses http/https protocol to communicate through port 443. If firewall settings prevent http/https communication through port 443, reconfigure the firewall to allow this traffic. Port 443 is the standard port for https communication.

The Batch Processor software uses the http/https protocol to communicate through port 444. If firewall settings prevent http/https communication through port 444, reconfigure the firewall to allow this traffic. See Chapter 4, Batch Processing.

### 3.2.4 Infrastructure

Because the *PostalOne!* system is web-based, there are no infrastructure requirements other than the hardware and software previously discussed.

### 3.2.5 Information/Manuals

To ensure that you accurately configure your Mail.dat files, use the *Mail.dat File Specification*, Version 02-2 and the *Mail.dat Users' Guide*, the two key documents from IDEAlliance. As a nonprofit organization, IDEAlliance charges a nominal fee for the file specification. The fee covers the costs of changes, updates, printing, and notifications to customers. If you do not have a current copy of the specification, contact IDEAlliance at (703) 837-1088 or online at <http://www.idealliance.org/maildat/>. You can download the *Mail.dat Users' Guide* from the website.

IDEAlliance  
c/o Mail.dat® V02-2  
100 Daingerfield Rd., Fourth Floor  
Alexandria, VA 22314

### 3.2.6 User License Code

To participate in the *PostalOne!* program, obtain a valid User License Code (ULC) from IDEAlliance, who assigns a ULC to each Mail.dat user. The ULC uniquely identifies you to the Postal Service. Customers who generate, update, or pass along Mail.dat files for subsequent use must have a ULC.

### 3.2.7 Security Certificates

The *PostalOne!* system uses Secure Sockets Layer (SSL) Version 3.0 to transfer files safely over the Internet. SSL is a secure enhancement to the standard Transmission Control Protocol/Internet Protocol (TCP/IP). It uses a combination of cryptographic processes to authenticate the host computers, and to encrypt and decrypt data transferred between them.

If you use the Manual mode file transfer process, the pre-installed VeriSign® client SSL certificate (included with the web browser) secures the file transmissions. No user setup is required for this process other than using the required web browser.

If you use the Batch mode file transfer process, your company must have a SSL certificate installed and set up. For information about security certificates, see Section 4.1.6, Digital Certificates/Security.

## 3.3 Configuring Mailing Files for Processing

This section provides details about the *Mail.dat File Specification* as they relate to data elements used by the *PostalOne!* system. Specifically, it covers required files and fields, typical problems, sending job updates, and file processing. We strongly recommend that you acquire the *Mail.dat File Specification* and the *Mail.dat Users' Guide* from IDEAlliance. For more information about the obtaining these, see Section 3.2.5, Information/Manuals.

### 3.3.1 Required Files and Fields

A Mail.dat job sent to the *PostalOne!* system consists of, at most, nineteen files. Each file in the set sent for a job consists of a different record type. Key fields within the records provide linkage from one file to another, thus creating a hierarchical relationship between the files. The *PostalOne!* system uses the Mail.dat standards as specified within the Mail.dat specification. In addition, the *PostalOne!* system specifically requires certain fields—these are marked in the specification with two asterisks (\*\*).

For a listing of the files available in Mail.dat, see Appendix A. Mail.dat File Definitions.

### 3.3.2 Typical File Formatting Problems

As more customers begin to utilize the *PostalOne!* system, we have been able to identify several common errors made in formatting fields. The more common errors are:

1. The optional date fields defined in the *Mail.dat File Specification* are often filled with zeros. The system validates all optional fields containing information and does not consider zeros to be acceptable date values.
2. The file/record level status flags are not used consistently. The system requires that all file/record level flags be consistent and does not allow any mixed Mail.dat transactions. For example, for an original Mail.dat transaction, all file level status flags should be set to "O".
3. You attempt to transfer a file but it fails because of an "Invalid Facility ZIP+4" in the segment record. When looking at the actual data, the file shows all of the data in the correct positions according to the *Mail.dat File Specification*. However, the software being used has written a blank record in the CR/LF pair causing the data in the Segment file to be off by two characters.

Solution: Delete the blank record and resave the file; the data will be in the proper positions within the file.

4. You used the File Validator to validate a Mail.dat file and it validated successfully. However, when attempting to transfer that Mail.dat file, it fails due to an "Invalid Facility ZIP+4" error.

This scenario can be confusing because the file passed validation. When validating the file, the File Validator does not transfer the file, nor does it compare the data to what is in your customer profile in the *PostalOne!* database. All of the data in the files may be in the right place but if the file transfer fails, the data in the required fields are not valid.

The problem is that the Mailing Facility ZIP+4 that is in position 138-146 in the segment file does not match the site's Mailing Facility ZIP+4 found in the *PostalOne!* database.

5. I/O Errors "Unable to connect to server." You can access the Internet and validate files, but when trying to transfer a file, you receive an I/O exception error.

There are a few situations that could cause this. Your inter-company network may allow read-only access to files. If you have your Mail.dat files saved on a network drive, the Transfer Applet is not being permitted to get the files and transmit those due to company firewall settings, inter-company network access, etc.

Solution: Create a folder on your local machine's hard drive, called "Mail.dat" for example, and save in that folder any Mail.dat files to be transferred to the *PostalOne!* system. Use the Transfer Applet and get the files from the new folder (i.e., Mail.dat). This way, the Applet does not contend with the network firewall issues.

6. I/O Errors "Unable to connect to server." You are transferring Mail.dat files from a folder on the local machine's hard drive (i.e., no network involvement, as described in the previous item) and still get an I/O exception error. Reasons:
  - a. You may have lost connectivity to the Internet during the process.
  - b. The *PostalOne!* database could be temporarily off-line.
  - c. It may be software related (an old browser or the user answered "No" to the security warning when trying to use the Transfer Applet).
7. "Trying to Reinsert an Original File." You have sent a specific Mail.dat job and it transferred successfully; however, when you try to resend the file, the *PostalOne!* database fails the transfer because the file has already been transmitted under that specific Job ID.

When transferring Mail.dat files to the *PostalOne!* database, remember that once an original Mail.dat was transferred successfully, you can only send subsequent updates to that job. For details about updates, see Section 3.3.3, Sending Job Updates.

To send an update to an existing Job in the system, make sure that all File Status flags in each file being transmitted reflect a "U" for update. All subsequent files included in the transfer of an update should include a "U" in the File Status flag field. For example:

- a. The Segment File (.seg) Status Flag, in position 187, should contain a "U" if it is an update.
  - b. The Mailpiece Unit (.mpu) Status Flag, in position 128, should contain a "U" if it is an update.
8. If the files were validated and transferred but not accepted by the database and you receive "Internal System Error," there may be null characters in the file between positions 292-301. The database will not accept a file with null characters. To spot the null characters, open the Mail.dat file using the Textpad utility. The null characters show up as black squares. Delete the black squares then resubmit the file.

### 3.3.3 Sending Job Updates

Large mailing jobs are often split into smaller production units, then produced and presented for acceptance incrementally (along with supporting documentation) over several days or even weeks.

Specific details of a mailing job may change. For example, a piece weight may have changed or an initial estimated weight becomes finalized when the mail is produced. Also, in-process mailing jobs may be canceled in whole or in part for a variety of reasons. The *PostalOne!* system, via the *Mail.dat File Specification*, accommodates these and other tasks and provides business mailers the means to communicate them to the Postal Service.

The scenarios such as those described above are handled as updates to the original Mail.dat file initially sent to the system. The list below includes the general guidelines for sending updated Mail.dat files:

- The system must receive an original Mail.dat file before any update can be received and processed.
- Mailers can send multiple updates to a mailing job, as long as they observe the rules established by the Mail.dat File Specification and those of the system.
- For mailers with an Optional Procedure (OP) mailing system, 100% of the containers in a mailing job must eventually be accounted for when a job is updated.

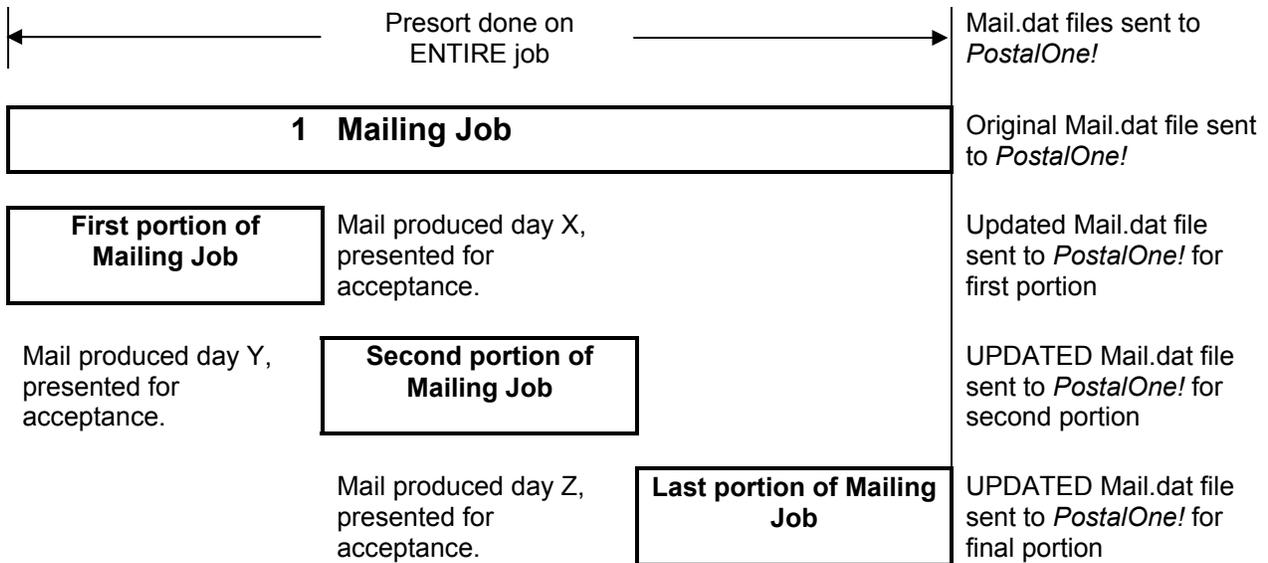
The table below lists the files a mailer would commonly include in original and update transfers.

**Table 3-2. Files Included in Original and Update Transfers**

Original Mail.dat File	Update
*Header	*Header
*Segment	*Segment
*Mailpiece Unit	Mailpiece Unit
*MPU/C Relationship	Component
*Component	Container Summary
*Container Summary	Container Quantity
*Container Quantity	
*Package Quantity	

\*Required files

Following is a common update scenario to an original Mail.dat file: The mailer creates an ORIGINAL Mail.dat mailing data file after presort processing a large job and sends that file to the *PostalOne!* system. For each portion of the job, the mailer produces the incremental portion, presents it for acceptance, and sends an UPDATE to the Mail.dat file. For the portion of the mail that is presented for acceptance that it is 'ready to pay,' a postage statement is created. This scenario is illustrated in Figure 3-2 below.



**Figure 3-2. Sending Job Updates to the *PostalOne!* System**

**NOTE:** Mailers can indicate that a portion or entire mailing job is ready for payment in an Original mailing file. If an entire job is ready to pay and will be presented for acceptance, an Update is not necessary.

### 3.3.3.1 File/Record Level Status

Mailers using the system must use file/record level status flags for all Mail.dat 02-2 files (history records can be of version 02-1). The table below lists the file/record level status flags allowed by the system.

**Table 3-3. File and Record Level Status Values**

Mail.dat Transaction Type	File Level Status	Record Level Status	Required Files (Minimum)
'O'iginal	All flags must be set to 'O'.	All flags must be set to 'O'.	HDR, SEG, MPU, MCR, CPT, CSM, CQT, PQT
'D'elete	All flags must be set to 'D'.	All flags must be set to 'D'.	HDR (alone), SEG and HDR (combined) <sup>1</sup>
'R'eplace	Not supported by the <i>PostalOne!</i> system.	Not supported by the <i>PostalOne!</i> system.	Not supported by the <i>PostalOne!</i> system.
'C'hange	All flags must be set to 'C'.	All flags must be set to either 'I' or 'U'.	HDR, SEG
'U'pdate	All flags must be set to 'U'.	All flags must be set to 'U'.	HDR, SEG

<sup>1</sup> The *PostalOne!* team strongly recommends that all delete transactions contain segment records to avoid accidental mass cancellations.

### 3.3.3.2 Rules for Sending Updates

When sending an update transmission, mailers participating in the *PostalOne!* program must follow these rules and requirements:

1. The Header file must contain a "U" in the appropriate status field.

- The *PostalOne!* system validates all container status updates and fails transactions that violate the rules in the following table, which contains the valid container status values to which a given container can be changed. If updating the Container Summary Record, position 172 must reflect the appropriate status and have one of the following allowable values:

**Table 3-4. Allowable Container Status Values**

Container Status	Allowable Changes	Notes
C		Cannot be updated
O	P, R, or C	
P	R or C	Can be updated only once per container job.
R	X or T	

- Original file transfers cannot have a "P" value.
- The "P" and "R" values can be transferred only once for a single container. The file fails if a second "P" or "R" is sent.
- The only way to update a container marked "R" is to use the "T" value to update the reservation number and container barcode information. A "T" value is accepted only after an "R" has been sent because no transportation update is necessary until the indication of payment. You can update transportation information only after it has been marked Ready to pay ("R").
- Updates to weight are reflected only in a postage statement with a "P" or an "R" value. If an update to weight is received within any other update, it is logged, but no recalculation of the postage statement occurs until a "P" or an "R" value is received.
- Only one estimated postage statement is generated for an entire mailing job, unless more than one actual postage statement is required for the mailing. Original job files cannot generate any preliminary statements, or group containers by date; to perform these functions, a separate update is required.
- Individual postage statements are generated for each unique Permit/USPS Pub number.

### 3.3.3.3 Commands

#### The Delete Transaction

You can close a mailing job (i.e., containers canceled) by sending a "delete" file. The *PostalOne!* application does not delete data, but instead, uses the delete command to signal a completion of the job. That is, the product, for whatever reason, is no longer going to mail.

A delete file is composed of a header file and an optional segment file. When only a header file is included, the *PostalOne!* system cancels all jobs for all plants (i.e., facilities) with the same Job ID and Provider Code.

When the segment file is included, you can specify individual facilities (using the Verification Facility ZIP+4 field). When the Verification Facility ZIP+4 is available, the job can be closed at a specific facility (if this job spans more than one facility). **NOTE:** Verification Facility ZIP+4 is contained in the segment file.

**NOTE:** To avoid accidental mass cancellations, the *PostalOne!* team strongly recommends that you include segment records in all delete transactions. Doing so prevents the following sample scenario:

A company (Provider Code) prepares two different jobs with the same Job ID at different plants (two distinct Verification Facilities ZIP+4 Codes). If the company then submits a delete transaction without a segment file, they cancel containers for both jobs.

## The Change Transaction

If the File Status fields in the header file for a Mail.dat job are set to "C" the job is considered a "Change" job. Per the *Mail.dat File Specification*, available change actions are 'I'nsert and 'U'pdate, based on the values contained in the record level status flags. Although the specification includes support for the 'D'elete flag, the *PostalOne!* system supports only the "I", "O" and "U" flags. Support for the "D" flag is pending.

## Replace Transactions

A Mail.dat job may be replaced by sending a 'delete' transaction (see above) then resubmitting the same job with a different Job ID.

### 3.3.4 File Processing Overview

When users select and send files manually, the *PostalOne!* system processes the files in the order the user selected them. When selecting multiple update files for transfer, users must select only those update files for which an original file has already been sent.

The batch file transfer module processes jobs in first-in-first-out (FIFO) order based on the date-time stamp on the Mail.dat files. However, this does not guarantee that the system completes jobs with different Job IDs in the original transfer order. For example, when a large job is transferred first and then followed by a smaller job with a different Job ID, the smaller job usually finishes first. This is because the file transfer module can process several jobs at a time, and the larger job takes longer to process.

The *PostalOne!* system Electronic Data Exchange process is detailed here and illustrated in Figure 3-3 below.

1. The customer transfers Mail.dat files to the *PostalOne!* system in one of two ways:
  - a. The Transfer Applet client, available on the *PostalOne!* website, manually transfers Mail.dat files to the *PostalOne!* File Upload Server.
  - b. The Batch Processor application, when scheduled, automatically transfers Mail.dat files to the *PostalOne!* File Upload Server using password protection and Secure Socket Layer (SSL) encryption. The File Upload Server then returns a Receipt file to the client machine.
2. The File Upload Server inserts the Mail.dat files into the *PostalOne!* database.
3. Various reports (including the postage statements) are made available from the *PostalOne!* web server. **NOTE:** Postage statements for Mail.dat files larger than 30MB are not immediately available, but will be processed within 24 hours.

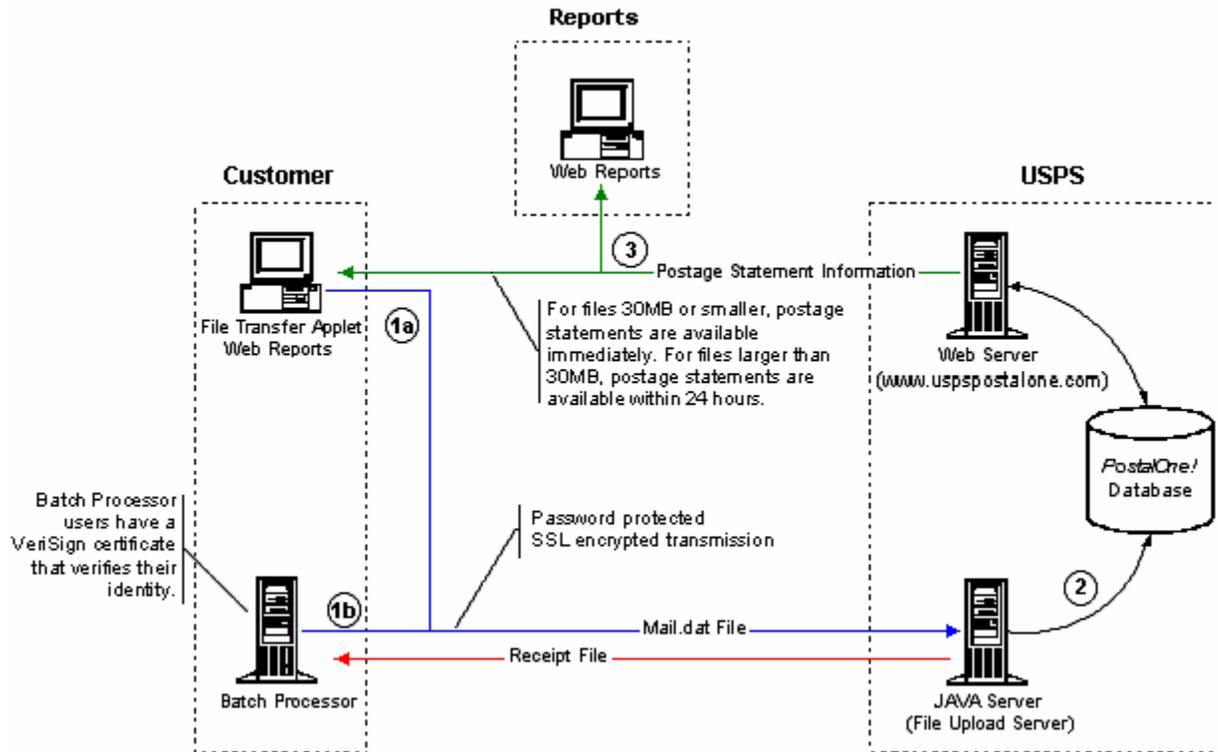


Figure 3-3. Mail.dat File Transfer Process

### 3.4 Transferring Files

When mailers send files to the *PostalOne!* system, the file transfer software performs a high-level file structure verification (called *file validation*) by checking the record counts for Mail.dat files provided in the Header record against the actual records present in the attached files. It also checks that all required fields exist. File Transfer fails the job if a required field is missing. It validates key field relationships between one or more records of files and verifies the structural and hierarchical integrity of the entire job by validating the proper existence of all predefined parent-child relationships.

To increase the likelihood of files transferring successfully, use File Validator to verify Mail.dat files before transferring them. The tool checks the validity of the data elements in the file and records error messages and warnings to the user's hard drive. To use the File Validator, under Tools, click **File Validator**. This is discussed in detail in the next section.

After the mailing passes validation, you are ready to transfer it. For information on this, see Section 3.4.2, Performing the File Transfer.

#### 3.4.1 Validating Files

The File Validator checks your Mail.dat files to ensure that all *required* fields are populated and contain the *correct character format*. This section explains how to use the File Validator feature.

**NOTE:** Before using the File Validator, you should verify that the Physical Memory (RAM) available on your machine meets the recommendations listed in the table below:

**Table 3-5. RAM Requirements Based on Net Job Size**

Net Size of Mail.dat Job (MB) *	Recommended Physical RAM
0 – 10	64 MB
11 – 25	128 MB
26 – 50	256 MB

\* The net size of a Mail.dat job is calculated as the "total size of all files" associated with a job, minus the size of the PQT, SNR, and SPR files.

**NOTE:** These memory requirements are for a single thread; when sending multiple simultaneous jobs, use the sum of all net sizes to estimate memory requirements.

To validate your Mail.dat files:

1. A special test system has been set up for new participants. It is located at <https://cf6.uspspostalone.com/postal1>. This tool was created to provide a safe environment for testing files and to help you learn about the *PostalOne!* system.
2. To open the File Validator page, on the left menu bar of the logon page, click **File Validator**. If you see a Security Warning pop-up box, click **Yes** to load the File Validator on your system as a temporary file. If you click **No** you will not load it or be able to use the File Validator.
3. The "All Folders" pane displays your computer's local drives (usually A, C, and D). From the list, click the plus sign (+) next to the drive on which your job is saved.
4. Scroll up or down to locate the folder in which your job is saved.
5. Select the proper folder by clicking the folder name. The "File Contents of" pane displays the job header files.

**NOTE:** If the folder containing your job is in a subfolder, click the plus sign (+) next to the main folder to display the subfolders. From the subfolders, locate and select the appropriate folder.

6. In the "File Contents of" pane, select the job header file you want to validate.  
**NOTE:** To validate multiple job header files simultaneously, hold down the CTRL key while you select the job files.
7. Before you validate the files, you may wish to change the validation.log file's name and location by clicking **Validation Log File**. By default, the File Validator writes the results to  
C:\Validation.log.
8. Click **Validate File(s)** to begin the validation process.
9. If your job passes the file validation process, a message indicating that the validation was successful appears. If your job fails the validation process, an error message appears. For more information about examining the validation log file, see Section 3.4.1.3, Validation Log Errors.
10. Click **OK** to close the message.

### 3.4.1.1 Validation Constraints

The following relationship constraints are validated for Mail.dat 02-2:

1. To successfully pass the Mail.dat validation, the *Mail.dat Presentation Category* field in the Header record must be populated with one of the data values listed in the table below:

**Table 3-6. Presentation Category Values Required to Pass Validation**

Data Value	Value Definition	Rules of Validation
P	Conventional Presort	<p>If Mail.dat Presentation Category is 'P', the following files must be present to successfully pass validation.</p> <ul style="list-style-type: none"> <li>• Header</li> <li>• Segment</li> <li>• MPU</li> <li>• MPU/C Relationship</li> <li>• Component</li> <li>• Container Summary</li> <li>• Container Quantity</li> <li>• Package Quantity - Required only if there is no Single-Piece file, .SPR</li> <li>• Single-Piece Record – Required only if there is no Package Quantity file, .PQT</li> </ul>
M	MLOCR	<p>If Mail.dat Presentation Category is 'M', the following files must be present to successfully pass validation.</p> <ul style="list-style-type: none"> <li>• Header</li> <li>• Segment</li> <li>• MPU</li> <li>• MPU/C Relationship</li> <li>• Component</li> <li>• Container Summary</li> <li>• Container Quantity</li> </ul>
I	Manifest Individual	<p>If Mail.dat Presentation Category is 'I', the following files must be present to successfully pass validation.</p> <ul style="list-style-type: none"> <li>• Header</li> <li>• Segment</li> <li>• Component</li> <li>• Container Summary</li> <li>• Manifest Individual</li> </ul>
S	Manifest Summary	<p>If Mail.dat Presentation Category is 'S', the following files must be present to successfully pass validation.</p> <ul style="list-style-type: none"> <li>• Header</li> <li>• Segment</li> <li>• Component</li> <li>• Manifest Summary</li> </ul>

Data Value	Value Definition	Rules of Validation
N	Single-Piece	<p>If Mail.dat Presentation Category is 'N', the following files must be present to successfully pass validation:</p> <p>Set A:</p> <ul style="list-style-type: none"> <li>• Header</li> <li>• Segment</li> <li>• Component</li> <li>• Manifest Individual</li> <li>• Container Summary</li> </ul> <p>- OR -</p> <p>Set B:</p> <ul style="list-style-type: none"> <li>• Header</li> <li>• Segment</li> <li>• MPU</li> <li>• MPU/C Relationship</li> <li>• Component</li> <li>• Container Summary</li> <li>• Container Quantity</li> <li>• Single-Piece Record</li> </ul>

2. The Mail.dat file structure is validated for the different Mail.dat transaction types, as follows:
  - The file level status flags must be consistent for a given Mail.dat transaction. For example, for an "original" transaction, all file level status flags in the header record must be set to "O". If some optional files are not included in the "original" transaction, the record count for those files must be 0 (zero). Similarly, for "update" or "change" or "delete" transactions, all file level status flags must be set to "U" or "C" or "D" respectively.
  - "Original" transactions – For original Mail.dat transactions, the Mail.dat file structure is verified based on the presentation category, as described above.
  - "Update" transactions – For all update transactions, the Header and the Segment files are required. All other files are optional.
  - "Change" transactions – For change transactions, the Header and the Segment files are required. All other files are optional.
  - "Delete" transactions – Two types of delete transactions are supported by the *PostalOne!* system. To delete a complete job (including all segments of the job), only a Header file can be sent. To delete specific segments in a job, a Header and a Segment file must be sent identifying the specific Verification Facility ZIP+4s to be deleted.
3. All key field relationships between the different records of the Mail.dat file are validated, based on the Presentation Category.
4. All Mail.dat records are validated to ensure that no duplicate records exist based on the key field combinations.
5. Mail.dat records are validated to ensure that all required children are present, for any given parent record of a Mail.dat job.
6. If the 'Sibling Container' field in the CSM record is set to 'Y', the following fields must have values:
  - Job ID
  - Segment ID
  - Container ID
  - Sibling Container Indicator
  - Sibling Container Reference ID

**NOTE:** All other fields in this CSM record **MUST** be left blank.

7. The Parent/Sibling Container relationship is validated. If a Sibling Container record exists, its parent must be present.
8. If the WSR file is used in the Mail.dat transmission, the 'Co-Palletization Code' field in the MPU record must be present.
9. If the 'Rate Category' field in the PQT record is set to 'A' (Saturation–ECR), the corresponding WSR record must be present.
10. All Component (CPT) records must have a value in the 'Permit ZIP+4/Postal Code' field. The Permit ZIP+4 value must be the ZIP Code associated with the post office where the permit is held.
11. Mail.dat files are validated for the reservation number in the CSM file according to the following:  
 The reservation number must consist of EXACTLY 12 characters, including spaces, and must be left-justified. The reservation number consists of the following three fields:
  - The first field begins at position one and must contain three to five alphanumeric characters, left-justified. If the data is less than five characters in length, blank spaces or hyphens must be used as padding.
  - The second field begins at position six and must consist of four numeric characters. The characters must correspond to a valid date in the form of MMDD, where MM is a two-digit month and DD is a two-digit date. LEADING ZEROS ARE REQUIRED for months or days less than 10.
  - The third field begins at position 10 and can consist of three alphanumeric characters (punctuation and space characters are not allowed). LEADING ZEROS ARE REQUIRED for months or days less than 10.
 The three fields of the reservation number discussed above correspond to the following:
  - Field One – the facility identifier
  - Field Two – the appointment date
  - Field Three – a sequence number or the text "std" (for standing appointment) or "sby" (for standby appointment)
12. Permit ZIP+4 – The *PostalOne!* application requires the "Permit ZIP+4" field as part of the information necessary to uniquely identify a Permit number. For the *PostalOne!* system to complete an end-to-end transaction, it is necessary to have the Permit ZIP+4 information with *all* postage transactions.

### 3.4.1.2 Relationship Constraints Description

This section describes every relationship constraint that is validated by the Phase III validation module, based on the different Mail.dat presentation categories. To determine which files are required for the various Mail.dat 'Conventional Presort' transactions, see Section 3.3.3, Sending Job Updates.

**NOTE:** Only the 'Conventional Presort' Mail.dat presentation category is currently implemented and tested in the *PostalOne!* system. All other presentation categories are currently supported by the file transfer module, but have not been tested as part of the complete *PostalOne!* system.

**HDR:** (required for all presentation categories)

- All header records present in a 'HDR' file must belong to a single job.
- There must be only one 'current' header record present for a job, in a 'HDR' file.
- There can be none, or any number of 'history' records present in a 'HDR' file for a given job. The header history record sequence number must be unique for each history header record.

**SEG:** (required for all presentation categories)

- There must be at least one or more segment record(s) present in the 'SEG' file for the unique Job ID present in the header file.

- No duplicate segment records should be present in the 'SEG' file considering the key fields of this record.

**CPT:** (required for all presentation categories)

- There must be at least one or more component record(s) present in the 'CPT' file for the unique Job ID present in the header file.
- No duplicate component records should be present in the 'CPT' file considering the key fields of this record.

**MPU:** (required only for 'Conventional Presort' and 'MLOCR' presentation categories)

- There must be at least one or more mailpiece unit record(s) present in the 'MPU' file for each segment record present in the segment file.
- No duplicate mailpiece unit records should be present in the 'MPU' file considering the key fields of this record.

**MCR:** (required only for 'Conventional Presort' and 'MLOCR' presentation categories)

- There must be at least one or more mpu/component relationship record(s) present in the 'MCR' file for each mailpiece unit record present in the MPU file.
- There must be at least one or more mpu/component relationship record(s) present in the 'MCR' file for each component record present in the CPT file.
- No duplicate mpu/component relationship records should be present in the 'MCR' file considering the key fields of this record.

**PAR:** (optional for all presentation categories)

- Because the 'PAR' records are optional, there can be one or more postage adjustment record(s) present in the 'PAR' file for each mailpiece unit record present in the MPU file.
- Because the 'PAR' records are optional, there can be exactly one postage adjustment record present in the 'PAR' file for each component record present in the CPT file.
- No duplicate postage adjustment records should be present in the 'PAR' file considering the key fields of this record.

**MSR:** (required only for 'Manifest Summary' presentation category)

- For the 'Manifest Summary' presentation category, there must be one or more manifest summary record(s) present in the 'MSR' file for each segment record present in the SEG file.
- For all other presentation categories, there can be one or more manifest summary record(s) present in the 'MSR' file for each segment record present in the SEG file.
- No duplicate manifest summary records should be present in the 'MSR' file considering the key fields of this record.

**WSR:** (optional for all presentation categories)

- Because the 'WSR' records are optional, there can be one or more walk sequence record(s) present in the 'WSR' file for each segment record present in the SEG file.
- Because the 'WSR' records are optional, there can be exactly one walk sequence record present in the 'WSR' file for each package quantity record present in the PQT file.
- No duplicate walk sequence records should be present in the 'WSR' file considering the key fields of this record.

**CSM:** (only optional for 'Manifest Summary' presentation category)

- For the 'Manifest Summary' presentation category, there can be one or more container summary record(s) present in the 'CSM' file for the unique Job ID present in the header file.
- For all other presentation categories, there must be at least one or more container summary record(s) present in the 'CSM' file for the unique Job ID present in the header file.
- No duplicate container summary records should be present in the 'CSM' file considering the key fields of this record.

**MIR:** (required only for 'Manifest Individual' presentation category)

- For the 'Manifest Individual' presentation category, there must be one or more manifest individual record(s) present in the 'MIR' file for each container summary record present in the CSM file.
- For all other presentation categories, there can be one or more manifest individual record(s) present in the 'MIR' file for each container summary record present in the CSM file.
- If the 'MIR' records are used, there can be exactly one manifest individual record present in the 'MIR' file for each special fees/chrng record present in the SFR file.
- No duplicate manifest individual records should be present in the 'MIR' file considering the key fields of this record.

**CLR:** (optional for all presentation categories)

- Because the 'CLR' records are optional, there can be exactly one container label record present in the 'CLR' file for each container summary record present in the CSM file.
- No duplicate container label records should be present in the 'CLR' file considering the key fields of this record.

**ICL:** (optional for all presentation categories)

- Because the 'ICL' records are optional, there can be exactly one international container label record present in the 'ICL' file for each container summary record present in the CSM file.
- No duplicate international container label records should be present in the 'ICL' file considering the key fields of this record.

**CQT:** (required only for 'Conventional Presort' and 'MLOCR' presentation categories)

- There must be at least one or more container quantity record(s) present in the 'CQT' file for each container summary record present in the CSM file.
- No duplicate container quantity records should be present in the 'CQT' file considering the key fields of this record.

**PLR:** (optional for all presentation categories)

- Because the 'PLR' records are optional, there can be one or more package label record(s) present in the 'PLR' file for each container summary record present in the CSM file.
- No duplicate package label records should be present in the 'PLR' file considering the key fields of this record.

**PQT:** (required only for 'Conventional Presort' if no SPR records are used)

- If a single-piece record is available for every container quantity record in a job, the package quantity records are optional. There can be one or more package quantity record(s) present in the 'PQT' file for each container quantity record present in the CQT file.

- If a single-piece record is not available for every container quantity record in a job, the package quantity records are required. There must be one or more package quantity record(s) present in the 'PQT' file for each container quantity record present in the CQT file.
- If the package label records are used, there can be one or more package quantity record(s) present in the 'PQT' file for each package label record present in the PLR file.
- No duplicate package quantity records should be present in the 'PQT' file considering the key fields of this record.
- All PQT records with the Saturation – ECR Rate Category (A) require one or more associated Walk Sequence (WSR) records or the job will fail validation.

**SPR:** (required only for 'Conventional Presort' if no PQT records are used)

- If a package quantity record is available for every container quantity record in a job, the single-piece records are optional. There can be one or more single-piece record(s) present in the 'SPR' file for each container quantity record present in the CQT file. Also, there can be one or more single-piece record(s) present in the 'SPR' file for each package quantity record present in the PQT file.
- If a package quantity record is not available for every container quantity record in a job, the single-piece records are required. There must be one or more single-piece record(s) present in the 'SPR' file for each container quantity record present in the CQT file
- No duplicate single-piece records should be present in the 'SPR' file considering the key fields of this record.

**SNR:** (optional for all presentation categories)

- Because the 'SNR' records are optional, there can be one or more seed name record(s) present in the 'SNR' file for each package quantity record present in the PQT file.
- No duplicate seed name records should be present in the 'SNR' file considering the key fields of this record.

**SFR:** (optional for all presentation categories)

- Because the 'SFR' records are optional, there can be exactly one special fees/chrng record present in the 'SFR' file for each single-piece record present in the SPR file.
- No duplicate special fees/chrng records should be present in the 'SFR' file considering the key fields of this record.

### 3.4.1.3 Validation Log Errors

If your job fails the validation process, an error message appears. To determine why the validation failed, write down the error message, and then inspect the log file written by File Validator. If you have not changed the name and location of the Validation Log file (see Section 3.4.1, Validating Files), its default is `C:\Validation.log`.

Use a text editor or other tool to view the log file.

**NOTE:** The Validation Log file is a simple (flat) ASCII file. To view it, use a text editor, such as Microsoft® WordPad, available with most versions of Microsoft Windows® operating system. For example, if you use WordPad to view the Validation Log file, use these steps:

1. Click **Start** and select **Run**. The Run dialog box opens.
2. In the **Open** box, type **wordpad** and click **OK** to open the WordPad window.
3. From the **File** menu, select **Open** to open the Open dialog box.

4. Make sure that “C:” appears in the Look in drop-down menu. If it does not, select it from the drop-down menu.
5. In the **File name** box, type **validation.log**, then click **Open**. WordPad opens the file and displays error messages from the oldest to most recent.

Scroll through the list of error messages and write down the error message you received for your job. If the File Validation Log entry(s) indicates that your mailing job (Mail.dat file) did not pass validation, you will need to analyze the Mail.dat file. If you produced your mailing file using a third-party vendor’s software product, you will probably need to contact them for help in resolving the problem. If you produced the mailing file using ‘in-house’ software, you will probably need to analyze the file or contact technical resources to help you resolve the problem. Some customers use a third-party Mail.dat viewer to analyze their Mail.dat files to resolve problems. Other customers analyze the Mail.dat file using a text editor capable of counting lines and character positions.

For a complete listing of error messages that appear in the Validation.log file, see Appendix D. Validation Log File Error Messages.

### 3.4.2 Performing the File Transfer

Before transferring your jobs to the Postal Service, you should first validate your files. For more information, see Section 3.4.1, Validating Files.

To transfer Mail.dat job files manually:

1. On the left menu bar, click **File Transfer**. You will probably see a Security Warning message. If you do not see it, minimize or move the browser window—it may be hiding the message. Click **Yes**. If you do not click Yes, or if you click No, you cannot transfer files to the *PostalOne!* system. The File Transfer page displays the “All Folders” and “File Contents of” panes.
2. The “All Folders” pane on the left displays your computer's local hard drives (usually A, C and D). Click the plus sign (+) next to the drive on which your jobs are saved.
3. In the “All Folders” pane, scroll up or down to locate the folder in which your job is saved.
4. Select the proper folder by clicking the folder name. The “File Contents of” pane on the right displays the job header files.

**NOTE:** If the folder containing your job is in a subfolder, click the plus sign (+) next to the main folder to display the subfolders. From the subfolders, locate and select the appropriate folder.

5. In the “File Contents of” pane, select the header file of the job you want to transfer.  
**NOTE:** All files associated with a mailing job are transferred as a result of selecting the header file. To transfer multiple jobs simultaneously, hold down the CTRL key and click the header files of the jobs one at a time to select them.
6. Once you have selected the job header file(s), you can determine the location and name of the log file, just as you did when using the File Validator. To do this, click **Validation Log File**. By default, the File Validator writes the results to `C:\Validation.log`.
7. Click **Transfer file(s)** to begin the transfer. While the file is transferring, you see a blue status bar beneath the “File Contents of” pane indicating transfer progress.
8. After your job has transferred, a message indicating the transfer status appears. Click **OK** to close the message and continue working. If you are transferring multiple jobs, the transfer message appears after each job is transferred. When each message appears, click **OK** to continue working.  
**NOTE:** If you do not click **OK** to close the message, the remaining jobs *will* continue to transfer in the background.
9. To check the status of your transferred jobs, click **Transfer Summary**.

## Check File Transfer and Upload Status

To check transferred job and file upload status:

1. On the File Transfer page, click **Transfer Summary**. The Job Status page opens. It does not initially display recently transferred jobs. To display recently transferred jobs, click the **Refresh** button at the top-right of the table.

**NOTE:** In the Job Status column, jobs are reported as "successful," "In progress," or "failed." If your job is "failed," see Section 3.4.1.3, Validation Log Errors.

2. By default, the Job Status page displays the first eight jobs. To view the next eight jobs, click **Next 8** at the bottom of the page.

**NOTE:** To view more or fewer records per page, select a number from the Show records drop-down list.

3. To check file upload status, click a Job ID in the Jobs column or enter the Job ID in the **Search Job ID** box, then click **Search**.

The Job Detail page displays all files associated with the selected job, file upload date and time, and upload status.

**NOTE:** The Completed Job column will either list the date and time at which a file was successfully uploaded, or report the file upload as "Failed" or "In progress."

4. To view the status of a file's progress, click the file name in the File (Click for Errors) column. A message box displays error details.
5. To close the message and continue working, click **Close**.
6. If your mailing job was not transferred successfully, you should check the Transfer Summary and the Validation Log file.
7. If the Transfer Summary or Validation Log file entry(s) indicates that your mailing job (Mail.dat file) was not transferred successfully, you will need to analyze your Mail.dat file. If you produced the mailing file using a third-party vendor's software product, you will probably need to contact them for help in resolving the problem. If you produced the mailing file using 'in-house' software, you will probably need to analyze the file or contact technical resources to help you resolve the problem. Some customers use a third-party Mail.dat viewer to analyze their Mail.dat files to resolve problems. Other customers analyze the Mail.dat file using a text editor capable of counting lines and character positions.

## 4 Batch Processing

Batch processing allows you to schedule file transfers to take place automatically. Once batch processing is set up, you do not need to be present for the file transfer to take place. This capability requires additional infrastructure and configuration work.

When using the Batch Processor, the *PostalOne!* servers return file transfer/upload status feedback to the sender's workstation automatically. The feedback information (receipt file) is written to the sender's workstation in either a delimited ASCII text or XML format. You can load the receipt file into a spreadsheet or a database for storage and viewing. A receipt file is saved in the same directory that contains your mailing jobs. For a detailed explanation about the contents of a receipt file and associated record types and format, see Appendix B. Status Receipt File Layout.

## 4.1 Prerequisites for Batch Processing

### 4.1.1 Hardware

For batch file transfers using UNIX, the minimum workspace requirements are a workstation with a 266 MHz processor and 64 MB RAM. For batch file transfers using NT, the minimum workstation requirements are a Personal Computer (PC) with a Pentium II 266 MHz processor and 64 MB RAM. The exact RAM requirements vary depending on the size of the Mail.dat job. Table 4-1 details the RAM requirements.

**NOTE:** The Batch Processor can also be run on machines with HP-UX operating systems.

**Table 4-1. RAM Requirements Based on Net Job Size**

Net Size of Mail.dat Job (MB) *	Recommended Physical RAM
0 – 10	64 MB
11 – 25	128 MB
26 – 50	256 MB

\* The net size of a Mail.dat job is calculated as the "total size of all files" associated with a job, minus the size of the PQT, SNR, and SPR files.

**NOTE:** These memory requirements are for a single thread; when sending multiple simultaneous jobs, use the sum of all net sizes to estimate memory requirements.

### 4.1.2 Software

The following software is required to use the Batch Processing functionality:

1. Operating System – Microsoft Windows NT for PCs or Sun Solaris 2.6 for UNIX Workstations.
2. Java Software – Sun Microsystems Java Runtime Environment (JRE) or SDK, latest revision of Version 1.3.1 or greater.

This is available from Sun Microsystems, Inc. at <http://java.sun.com/products/archive/index.html>. Select Java 2 JRE or SDK, Version 1.3.1 or greater. Review and accept the download terms and conditions. Next, select the JRE or SDK that corresponds to your operating system configuration. Then select a "Typical" installation, and note the directory to which the Java software is installed (e.g., C:\program files\java\j2re1.3.1\bin).

3. Scheduling software – software that lets you schedule batch processing of your mailing files to run on the dates and at the times you require. The default scheduling utility for the operating system on the machine you use to send mailing files (e.g., cron/UNIX or "AT" task scheduler/Windows NT) may work.
4. Digital Certificate – get and install a digital certificate for each server that is used to send mail files. For more information, see Section 4.1.6, Digital Certificates/Security.

### 4.1.3 Network

The Batch Processor software uses the http/https protocol to communicate through port 444. If firewall settings prevent http/https communication through port 444, reconfigure the firewall to allow this traffic.

#### 4.1.4 Downloading and Configuring the NT Batch Processor

To download and configure the NT Batch Processor:

1. Launch Windows Explorer and in your root directory, create a *PostalOne!* Batch Processor installation directory. This is where you will place the batch processor software.  
**NOTE:** Give the folder a name that you will readily remember, such as: C:\MAILDAT.
2. Log on to the *PostalOne!* system, <http://www.uspspostalone.com/>.
3. On the left menu bar, click **Download Batch Processor** and download the following files into the installation directory:
  - WinZip file – Java archives that perform the batch processing
  - Parameters Files – Local host parameter for the batch processor
4. Extract the files from **daemon.zip**, using WinZip, into the *PostalOne!* Batch Processor installation directory (double-click the zip file and extract the contents). Once extracted, the directory should contain:
  - **daemon.jar** – Java archive file
  - **jsse.jar** – Java Secure Socket Extension (JSSE) Java archive file
  - **jcrt.jar** – JSSE Java archive file
  - **jnet.jar** – JSSE Java archive file
  - **client.config** – Client configuration file that specifies parameters to connect to the *PostalOne!* system
  - **postal1.ini** – Daemon initialization file that specifies local host parameters
5. Copy the three JSSE archive files to the extension library ( ..\lib\ext ) directory created when the Java JRE / SDK software was downloaded and installed:  
For example:
  - Java Runtime Environment - C:\Program Files\JavaSoft\JRE\1.3\lib\ext
  - Java 2 SDK - C:\jdk1.3\jre\lib\ext
6. Create a “Repository” subfolder in the *PostalOne!* batch processor installation directory.
7. This folder is a repository for your Mail.dat files. For example: C:\MAILDAT\Repository.
8. Create two subfolders in the new repository folder—one for successful transmissions and one for unsuccessful transmissions. For example: C:\MAILDAT\Repository\Successful and C:\MAILDAT\Repository\Unsuccessful. The batch processor deposits transmission files in these folders as each job is processed.
9. After you have created all three subfolders, edit the `Postal1.ini` file using Notepad, for example, and make the following modifications to the [Repository] section to reflect the directory structure you created above.

```
[Repository]
RepositoryLocation=C:\MAILDAT\REPOSITORY
SuccessfulRepositoryLocation= C:\MAILDAT\REPOSITORY\SUCCESSFUL
UnsuccessfulRepositoryLocation= C:\MAILDAT\REPOSITORY\UNSUCCESSFUL
```

**NOTE:** `Postal1.ini` is located in your batch processor installation directory (i.e. C:\MAILDAT).

The parameters required to efficiently use the *PostalOne!* system are provided in the table below.

**Table 4-2. Postal1.ini Parameters for NT**

Parameter Name	Valid Values	Default	Description
MaximumOpenThreads	1-50	4	Maximum threads that can be open at one time
RepositoryLocation	Any valid directory path.	C:\postal1\data	Location of Mail.dat files waiting to be processed
SuccessfulRepositoryLocation	Any valid directory path.	C:\postal1 \SUCCESSFUL	Location where the Mail.dat files are moved after they are successfully processed.
UnSuccessfulRepositoryLocation	Any valid directory path.	C:\postal1 \UNSUCCESSFUL	Location where the Mail.dat files are moved if the processing fails.
ReceiptFileRepositoryLocation	Any valid directory path.	C:\postal1 \ReceiptFile	Location where the receipt files are written by the application. This is the default for both status and postage statement receipts. However, it is used only for status receipts if the PSReceiptFileRepositoryLocation is set.
ReceiptFileName	Any valid file name.	Report	The name of the receipt files. Note that a unique three-digit sequence number is used to define the extension of the receipt files.
Delimiter	Any printable ASCII character	,	This character delimits all receipt file fields.
NumberOfValidationLogFiles	Any Numeric value. *	5	Number of validation log files that are written by the application before the first file is overwritten.
NumberOfServerLogFiles	Any Numeric value. *	5	Number of server log files that are written by the application before the first file is overwritten.
MaximumValidationLogLength	Any Numeric value. *	100000	The maximum size (in bytes) of a single validation log file.
MaximumServerLogLength	Any Numeric value. *	1000000	The maximum size (in bytes) of a single server log file.
SystemUsername	A valid <i>PostalOne!</i> account User Name.	Jdoe	A valid <i>PostalOne!</i> user name belonging to an active account that allows the account holder to transfer files.
SystemPassword	A valid <i>PostalOne!</i> account password.	Password	A valid <i>PostalOne!</i> password belonging to an active account that allows the account holder to transfer files.

Parameter Name	Valid Values	Default	Description
FileType	MAILDAT or DELCON	MAILDAT	Specifies the type of file that is processed by the application. MAILDAT for transferring Mail.dat files, and DELCON for transferring e-VS manifest files.
StatusReceiptFilename	Any valid file name.	receipt	Determines the name of status receipt file. The <i>PostalOne!</i> system appends a three-digit number to the name, which is incremented with each new receipt.
PostageStatementReceiptFilename	Any valid file name.	psreceipt	Determines the name of the postage statement receipt file. The <i>PostalOne!</i> system appends a three-digit number to the name, which is incremented with each new receipt.
PSReceiptFileRepositoryLocation	Any valid directory path.		Location where the postage statement receipt files (if set) are written by the application.
PSReceiptType	XML or DEL		Determines the format of the postage statement receipt file, either XML or character delimited (DEL).
StatusReceiptType	XML or DEL		Determines the format of the status receipt file, either XML or character delimited (DEL).

\* **NOTE:** Sufficient disk space should be available to store the number of specified files.

The parameter `MaximumSystemErrors` is no longer required in the `postal1.ini` file. If it is still there, you should remove it.

#### 4.1.5 Downloading and Configuring the UNIX Batch Processor

To download and configure the UNIX Batch Processor:

1. Create a *PostalOne!* Batch Processor installation directory. The batch processor will be located in this directory. For example: `/postal1/data`.
2. Log on to the *PostalOne!* system, <http://www.uspspostalone.com/>.
3. On the left menu bar, click **Download Batch Processor** and download the required files to your batch processor installation directory:
  - Archive file (`daemon.tar`) – Java archives that perform the batch processing
  - Parameter File (`postal1.ini`) – Local host parameters for the batch processor
4. Use tar to extract the files into the installation directory. Once extracted, the directory should contain:
  - **daemon.jar** – Java archive file
  - **jsse.jar** – Java Secure Socket Extension (JSSE) Java archive file
  - **jcrt.jar** – JSSE Java archive file
  - **jnet.jar** – JSSE Java archive file

- **client.config** – Client configuration file that specifies parameters for connecting to the *PostalOne!* system
  - **postal1.ini** – Daemon initialization file that specifies local host parameters
5. Copy the three JSSE archive files to the extension library directory ( ...\lib\ext ) created when the Java software was downloaded and installed.  
For example:
    - Java Runtime environment - j2re1\_3\_0/lib/ext
    - Java 2 SDK - j2se/jre/lib/ext
  6. Once the files are successfully copied or downloaded, create a “repository” subfolder in the installation directory. This folder is a repository for your Mail.dat files, for example:  
/postal1/repository.
  7. Create two subfolders of the new repository folder—one for successful transmissions and one for unsuccessful transmissions. The batch processor deposits transmission files in these folders as each job is processed, for example: /postal1/repository/successful and /postal1/repository/unsuccessful.
  8. After you have created all three subfolders, edit the Postal1.ini file and make the following modifications to the [Repository] section to reflect the directories you created:  
[Repository]  
RepositoryLocation=/postal1/repository  
SuccessfulRepositoryLocation=/postal1/repository/successful  
UnSuccessfulRepositoryLocation=/postal1/repository/unsuccessful

**NOTE:** *Postal1.ini* is located in your initial installation directory. The parameters required to efficiently use the *PostalOne!* system are provided in the table below.

**Table 4-3. Postal1.ini Parameters for UNIX**

Parameter Name	Valid Values	Default	Description
MaximumOpenThreads	1-50	4	Maximum threads that can be open at one time
RepositoryLocation	Any valid directory path.	C:\postal1\data	Location of Mail.dat files waiting to be processed
SuccessfulRepositoryLocation	Any valid directory path.	C:\postal1\SUCCESSFUL	Location where the Mail.dat files are moved after they are successfully processed.
UnSuccessfulRepositoryLocation	Any valid directory path.	C:\postal1\UNSUCCESSFUL	Location where the Mail.dat files are moved if the processing fails.
ReceiptFileRepositoryLocation	Any valid directory path.	C:\postal1\ReceiptFile	Location where the receipt files are written by the application.
ReceiptFileName	Any valid file name.	report	The name of the receipt files. Note that a unique three-digit sequence number is used to define the extension of the receipt files.
Delimiter	Any printable ASCII character	,	This character delimits all receipt file fields.

Parameter Name	Valid Values	Default	Description
NumberOfValidationLogFiles	Any Numeric value. *	5	Number of validation log files that are written by the application before the first file is overwritten.
NumberOfServerLogFiles	Any Numeric value. *	5	Number of server log files that are written by the application before the first file is overwritten.
MaximumValidationLogLength	Any Numeric value. *	100000	The maximum size (in bytes) of a single validation log file.
MaximumServerLogLength	Any Numeric value. *	1000000	The maximum size (in bytes) of a single server log file.
SystemUsername	A valid <i>PostalOne!</i> account User Name.	Jdoe	A valid <i>PostalOne!</i> user name belonging to an active account that allows the account holder to transfer files.
SystemPassword	A valid <i>PostalOne!</i> account password.	password	A valid <i>PostalOne!</i> password belonging to an active account that allows the account holder to transfer files.
FileType	MAILDAT or DELCON	MAILDAT	Specifies the type of file that is processed by the application. MAILDAT for transferring Mail.dat files, and DELCON for transferring e-VS manifest files.
StatusReceiptFilename	Any valid file name.	receipt	Determines the name of the status receipt file. The <i>PostalOne!</i> system appends a three-digit number to the name, which is incremented with each new receipt.
PostageStatementReceiptFilename	Any valid file name.	psreceipt	Determines the name of the postage statement receipt file. The <i>PostalOne!</i> system appends a three-digit number to the name, which is incremented with each new receipt.
PSReceiptFileRepositoryLocation	Any valid directory path.		Location where the postage statement receipt files (if set) are written by the application.
PSReceiptType	XML or DEL		Determines the format of the postage statement receipt file, either XML or character delimited (DEL).
StatusReceiptType	XML or DEL		Determines the format of the status receipt file, either XML or character delimited (DEL).

\* **NOTE:** Sufficient disk space should be available to store the number of specified files.

The parameter `MaximumSystemErrors` is no longer required in the `postall.ini` file. If it is still there, you should remove it.

## 4.1.6 Digital Certificates/Security

The *PostalOne!* system uses the Secure Sockets Layer (SSL) version 3.0 to transfer files securely over the Internet. SSL is a secure enhancement to the standard Transmission Control Protocol/Internet Protocol (TCP/IP). It uses a combination of cryptographic processes to authenticate the host computers and to encrypt and decrypt data transferred between them. To transfer files securely (Batch only) with the *PostalOne!* system, complete the following tasks to permit SSL to operate on your host system:

1. Create a Keystore
2. Create a Certificate Signing Request
3. Submit your CSR and procure a Digital Certificate
4. Add the Signed Certificate to your Keystore
5. Enable Encryption on the Batch Processor Client
6. Test your Signed Certificate
7. Test the Batch Processor Setup

Each of these steps is discussed in detail below.

### 4.1.6.1 Creating a Keystore

A *keystore* is a file that contains a digitally signed certificate. The Batch Processor file transfer system uses this certificate to verify the name of your host computer before accepting files from it.

To create a keystore, use the **keytool** utility that is part of the Java JRE/SDK software you downloaded earlier. You will need to know the following information to create a keystore:

- The *fully-qualified* domain name of your file transfer computer (i.e., ARLNVAUX002.usps.gov)

**NOTE:** The common name must match the domain name.

- The name of your organizational unit (i.e., Marketing)
- The name of your organization (i.e., United States Postal Service)

**NOTE:** The formal name must be alphanumeric characters (no @ or &, for example).

- The city, state or province, and two-character country code of your location.  
An example of creating a keystore follows: (**NOTE:** Use the directory where the keytool software utility resides.)

```
C:\(directory of keytool java software)> keytool -genkey -alias postalone -keystore p1keys -keyalg rsa
```

**NOTE:** You must specify **-keyalg rsa** when creating a keystore for use with the *PostalOne!* system. The *PostalOne!* system uses the RSA signature algorithm for signing certificates.

```
Enter keystore password: postal1 (for example)
```

```
What is your first and last name?
```

```
[Unknown]: ARLNVAUX002.usps.gov
```

```
What is the name of your organizational unit?
```

```
[Unknown]: Marketing
```

```
What is the name of your organization?
```

```
[Unknown]: United States Postal Service
```

```
What is the name of your city or locality?
```

```
[Unknown]: Arlington
```

```
What is the name of your state or province?
```

```
[Unknown]: Virginia
```

```
What is the two-letter country code for this unit?
```

```
[Unknown]: US
```

Is <CN=ARLNVAUX002.usps.gov, OU=PostalOne, O=United States Postal Service, L=Arlington, ST=Virginia,C=US> correct?[no]:

[Unknown]: y

Enter key password for <postalone>  
(RETURN if same as keystore password):

**NOTE:** Record your keystore password in a safe place. You will need it in subsequent steps.

#### 4.1.6.2 Creating Certificate Signing Requests

Use **keytool** to create your certificate signing request (CSR). **NOTE:** Use the directory where the keytool software utility resides.

```
C:\(directory of keytool java software)>Keytool -certreq -alias postalone - keystore p1keys - file p1req.txt
```

Enter keystore password: **postal1** (for example)

Your CSR is contained in the file p1req.txt.

#### 4.1.6.3 Submitting Your Certificate Signing Request

You must now submit your CSR for digital signing, i.e., procure a digital certificate. Currently, the *PostalOne!* system uses digital certificates issued by VeriSign. To obtain a certificate, go to <http://www.verisign.com/products/site>, and obtain a basic **40-bit SSL Secure Server ID** for each machine that will send batch files to the *PostalOne!* system.

**NOTE:** Your digital certificate generally expires after one (1) year. You need to renew it after that.

#### 4.1.6.4 Adding a Signed Certificate to a Keystore

After you submit a CSR to VeriSign, you will receive a digitally signed certificate. You must then add this signed certificate to your keystore. The file transfer system uses this signed certificate to verify the identity of your host computer before accepting any files from it.

Begin by getting the reply certificate from VeriSign and copying it into a text editor. It should look similar to the following example:

```
-----BEGIN CERTIFICATE-----
MIICdjCCAiACEBKVRKjE/nsXwEii3fN6k9AwDQYJKoZIhvcNAQEEBQAwgaxFjAU
BgNVBAoTDVZlcm1TaWduLCBJbmMxRzBFBG9NVBAStPnd3dy52ZXJpc2lnbi5jb20v
cmVwb3NpdG9yeS9UZXR0Q1BTIEluY29ycC4gQnkkgUmVmLiBmaWFiLiBMVEQuMUYw
RAYDVQQLZ1Gb3IgdG9yYVpZ24gYXV0aG9yaXplZCB0ZXN0aW5nIG9ubHkuIE5v
IGFzZ3V5YW5jZXMgKEMpVlMxOTk3MB4XDTAyMDMwNDAwMDAwMFoXDTAyMDMxODIz
NTk1OVowgY4xCzAJBgNVBAYTAlVTMREwDwYDVQQIEWhWaxJnaW5pYTESMBAGA1UE
BxQJQXJsaW5ndG9uMSUwIwYDVQQKFBxVbml0ZWQgU3RhdGVzIFBvc3RhbCBTZXJ2
aWNlMRIwEAYDVQQQLFAlQb3N0YWxPbmUxHTAbBgNVBAMUFEFSTE5WQVdaMTg0LlVT
UFMuR09WMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDGzthcIrSzSHFH13x1
ojzhPjacRUSH1vAuk8SZN1VVRrwi76dl4VqtsGpMD/YjmmMbN1fwgTFm6rfpMRXF
rWbQaMe9aB6EPZPwOhpEYdTY3Z9QpOwyKdu6amCwyQzUKkNV7sOeycSe+e9cW4JK
0Koxey14YvLkEu03DHGi0zQ0BwIDAQABMA0GCSqGSIb3DQEBAQUAA0EAv5bA+oCG
AzzhZsMic1S2O31iuVN3ppLrRU+DdCvi6otcpc0xUYVYjXFi83POzgjmdr0Rjt2g
QSWT9fCoL281gA==
-----END CERTIFICATE-----
```

Make sure that the certificate context (the text between the BEGIN CERTIFICATE line and the END CERTIFICATE line) has no spaces or return characters when copying and pasting. If there are any

spaces or return characters, delete them. **Put one and only one new line (i.e. carriage return) without any space after the END CERTIFICATE line. Then save the file (e.g., p1cert.txt).**

After you have saved the signed certificate, use **keytool** to import your certificate into your keystore.

**NOTE:** Use the directory where the keytool software utility resides.

```
C:\(directory of keytool java software)>keytool -import -trustcacerts -file p1cert.txt -alias postalone -keystore p1keys  
Enter keystore password: postal1 (for example)
```

#### **Certificate was added to keystore**

Now use keytool to verify that your signed certificate is in your keystore:

```
C:\postalone>keytool -list -keystore p1keys -alias postalone -v  
Enter keystore password: postal1 (for example)  
Alias name: postalone  
Creation date: Wed Oct 29 11:41:01 EDT 2003  
Entry type: trustedCertEntry
```

```
Owner: CN=ARLNVAUX002.usps.gov, OU= Marketing, O=United States Postal Service,  
L=Arlington, OID.2.5.4.17=94497, ST=VA, C=US  
Issuer: CN=USPS CA, OU=www.usps.com/CPS, O=United States Postal Service, C=US  
Serial number: 2895  
Valid from: Tue Oct 28 14:08:07 EDT 2003 until: Wed Oct 27 14:08:07 EDT 2004  
Certificate fingerprints:  
    MD5: 69:2C:04:E3:B7:F1:A9:1F:DE:FD:0E:36:A2:1B:FF:12  
    SHA1:E5:91:36:25:76:2B:CA:E0:DF:DD:3C:2B:91:E8:3B:97:B3:E6:8D:CE
```

**NOTE:** In the example above, the certificate is valid for one year, then it expires.

### **4.1.6.5 Enabling Encryption on the Batch Processor Client**

To enable the batch processor client secure file transfer feature, you must copy your keystore file to the directory in which you have installed the batch processor client software. You must also edit the `client.config` file. The `client.config` is already located in the directory in which you have previously installed the batch processor client software.

After copying your keystore file (for example, `p1keys`) to the directory containing the batch processor client, edit the `client.config` file. To enable encryption, you must set the **encryption** parameter to **true**. You must also supply the name of your **keystore** file and its corresponding **password**. When you are finished, these entries should look similar to the following example:

```
encryption = true  
keystore = p1keys  
password = postal1 (for example)
```

### **4.1.6.6 Testing Your Signed Certificate**

Your batch processor client software gives you the ability to test the installation of your certificate. This test is performed from the command prompt on your local file transfer host.

To test the installation of your signed certificate, make sure the **encryption** parameter of your `client.config` file is set to **true**. Then execute the following command. **NOTE:** Use the batch processor installation directory.

```
C:\(batch processor installation directory)>java -cp Daemon.jar Daemon -testlocal  
If your certificate is valid and is installed properly, you will see the following:  
Local Server Listening for Client Connection  
Response Code is : 200
```

Response Message is : OK  
This (Certificated) Local Connection test is SUCCESSFULL!

#### 4.1.6.7 Testing Batch Processor Setup

The batch processor client software also lets you test that the batch processor has been set up correctly and can communicate with the *PostalOne!* system. This test is run from the command prompt on your local file transfer host. **NOTE:** Use the batch processor installation directory.

```
C:\(batch processor installation directory)>java -cp Daemon.jar Daemon -testremote
```

#### 4.1.7 Scheduling Batch Jobs (Running the Batch Processor)

To have your jobs run automatically, use a scheduling utility/program that allows you to set the date and time the batch file transfer processor runs. Copy or move the mailing files into the repository directory before the batch job executes. To run the batch processor application, you invoke it at the command prompt. For example:

```
java -cp C:\<dir>\daemon.jar Daemon
```

Where **<dir>** is the directory containing the Java archive file (daemon.jar), the Daemon initialization file (postal1.ini), and client configuration file (client.config).

### 4.2 File Transfer Error Codes and Messages

When you transfer mailings, the system records error codes and descriptive messages. If you transferred the mailing using the manual process or the batch process, you can retrieve the error messages using the File Transfer page and selecting the **Transfer Summary** option.

Additionally, if you transferred the mailing using the batch process, error messages are stored in the folder/directory used as the installation folder/directory for batch file transfers.

The complete list of file error messages is in Appendix E. File Transfer Error Messages.

## 5 Customer Support

The *PostalOne!* program has a staffed Customer Care Center (help desk) to assist customers who may have questions or need assistance with a problem or technical issue with the *PostalOne!* system. The *PostalOne!* Customer Care Center is available Monday through Friday from 8:00 a.m. to 5:00 p.m., CST, and can be contacted via e-mail or telephone. You can also call the telephone number below for after-hours assistance with some issues. Our after-hours staff will forward unresolved issues to the Customer Care Center at the beginning of the next workday.

Customer Care Center Telephone	<b>(800) 522-9085</b>
Customer Care Center e-mail	<b>postalone@email.usps.gov</b>

If you experience any difficulties when using the *PostalOne!* system, have questions, or need more information about the *PostalOne!* system, contact the Customer Care Center. Your calls will be answered by the Customer Care Center in a priority order, with preference given to fully enrolled customers having a signed *PostalOne!* user agreement on file.

If you have comments or suggestions regarding this guide, e-mail the Customer Care Center.

## Appendix A. Mail.dat File Definitions

This appendix contains the names and definitions of each file used in Mail.dat. (Source: IDEAlliance's *Users' Guide for Mail.dat, Version 02-2*)

**Table A-1. Mail.dat Files and Definitions**

File	Definition
Header file (.HDR)	The Header file is an introduction to the entire Mail.dat. It identifies who created the file, what version of the IDEALLIANCE standard was used (ensuring compatibility with other users), and when the file was created. Most importantly, it identifies the total file record count for each file type. If the quantity fields in the Header record say there are a differing number of records than received, it is the first indication that the file is incomplete and, therefore, invalid. The presort software used, history of who has touched this Mail.dat, and other information is also in the Header.
Segment file (.SEG)	The Segment file is used to identify all of those addresses that are considered as a group within a presort. It separates parts of a mailing that require different processing. The specification notes:  In general, the fewer the segments in a Mail.dat, the better. It is only appropriate to create a unique segment when it is needed to separate part of a mailing for different processing. Segmenting should not be used to differentiate among entry points unless they will need to be processed in some fundamentally different fashion. Similarly, segmentation should not be used to create reporting categories from information that is otherwise available in the Mail.dat.  The Segment file identifies the class and characteristics of the mail preparation for each Segment.
Mailpiece Unit file (.MPU)	The Mailpiece Unit file contains the physical description of the whole mailpiece in terms of physical attributes, such as dimensions and ad percentages. Often, there is a single MPU within a Segment. A Mailpiece Unit ID cannot cross Segment; however, co-mingled mail, selectively inserted letters, and selectively bound catalogs/periodicals can have multiple Mailpiece Units associated with one segment. Another example of where there might be multiple Mailpiece Units in one segment is if multiple editions are produced in separate processes, but staged onto the same pallets on the backend in order to be presented as part of one mailing.
Component file (.CPT)	The Component ID file is used to define those parts of a Mailpiece Unit by the mail class for which they qualify. For example, a magazine is in the mail class of Periodicals. However, the publisher may decide to have a mailpiece that's ineligible for Periodicals rates to be attached to the front of the magazine. The ineligible mailpiece might be a Standard Mail™ piece. Each of these parts would be identified with a separate Component record that is linked to the appropriate MPU.
MPU/Component Relationship file (.MCR)	The MPU/Component Relationship file is a table relating the two variables. Like the MPU and Segments relationships, often there is a one-to-one relationship with an MPU and a Component. However, as described in the above file definition, there are exceptions. Periodicals with First-Class Mail attachments and Periodicals with Standard Mail enclosures are two examples.
Container Summary file (.CSM)	The Container Summary file identifies each container, its level, its destination, the point of entry and the contents in terms of rates, copies, pieces and weight. Pallets, sacks, and trays are containers. There will be one Container Summary per container.

File	Definition
Container Label file (optional) (.CLR)	The Container Label file contains the label information required for each container. Examples of container labels are pallet labels, tray labels, and sack labels.
International Container Label file (optional) (.ICL)	The International Container Label file container label file contains label information for each USPS international mail container. This file also has unique subsections applicable only to USPS international mail.
Container Quantity file (.CQT)	The Container Quantity file supplies quantitative details about the content of each container. It summarizes the contents of each container in terms of MPUs, zones, and rate categories. It also supplies copy and piece counts at this level of detail. Unlike the usually limited relationship of MPU/Segment, and Component /MPU, it is expected that there will be a great number of Container Quantity records linked to the same Container Summary.
Package Quantity file (optional) (.PQT)	The Package Quantity file presents a finer level of detail defining the quantity, rate, weight, and package destination for every package in a container. Although optional, this file is required if the application is to produce a standard USPS Qualification report and/or provide input to the USPS <i>PostalOne!</i> system.
Walk Sequence file (optional) (.WSR)	The Walk Sequence file provides detail for carrier route packages in a Walk Sequence mailing: package ZIP Code, carrier route number, walk sequence type, total stops made, and total possible stops.
Seed Name file (optional) (.SNR)	The Seed Name file provides the precise container, package, entry, version, etc. that applies to the respective Seed Names within the mailing. For those mailers who use seed addresses such as subscriber reporters or mail monitors, this file can be supplied by the presort provider to support communications regarding the total tracking program.
Package Label file (optional, Canadian only) (.PLR)	The Package Label file, like the Container Label file, is information to be used to print Canadian package labels (facing slips).
Ink Jet/Container Relationship file (optional) (.ICR)	The Ink Jet/Container Relationship file defines the specific location of each container on the associated ink jet tapes or transmitted files.
Single-piece file (optional) (.SPR)	The Single-piece file is used by those using “manifest” mailing procedures and by presort vendors pursuing the USPS PAVE certification.
Special Fees and Charges file (optional) (.SFR)	The Special Fees and Charges file can represent various fixed and weight-variable special fees and charges for any or all of the respective single-pieces within the mailing. Examples are Insured, Delivery Confirmation, and Signature Service.
Manifest Summary file (optional) (.MSR)	The Manifest Summary file is used for “manifest” mailings, where there is repetition of item characteristics, which lends itself to “summarized” presentation, rather than an individual SPR record for every piece.
Manifest Individual file (optional) (.MIR)	The Manifest Individual file is used for “manifest” mailings, especially suitable for parcels.
Postage Adjustment file (optional) (.PAR)	The Postage Adjustment file defines any postage adjustments required to be made, as compared to the anticipated production specifics as supplied within the Mail.dat. An example would be a penalty for too many unreadable barcodes.

## Appendix B. Status Receipt File Layout

This appendix contains details about the contents of a Status Receipt file and associated record types and format.

### B-1 Status Receipt File Content

The receipt file conforms to these specifications:

1. A receipt file shall contain one or more records represented by ASCII text. The text shall be in the subset of ASCII characters indicated by the decimal values 33 to 126.
2. A new line character or sequence of characters (this is platform dependent) shall terminate each record.
3. The records of a receipt file shall contain a predefined character that shall serve to distinguish its fields. The default character shall be a comma “,”.

**NOTE:** You can customize the ‘delimiter’ character by changing the Delimiter parameter in the `postal1.ini` file. For more information, see Section 4.1.4, Downloading and Configuring the NT Batch Processor or Section 4.1.5, Downloading and Configuring the UNIX Batch Processor.

4. The receipt file record may have various types. Only one record type has been specified at this point.
5. A record in a receipt file shall commence with a “record-type” field that shall serve in the record’s identification. Following the record-type field shall be various character-delimited fields following a predefined sequence.

### B-2 Record Types and Record Format

This section provides details about the receipt file’s record types and formats. For additional information about `postal1.ini` parameters, see the table in Sections 4.1.4, Downloading and Configuring the NT Batch Processor or 4.1.5, Downloading and Configuring the UNIX Batch Processor.

#### Record Type: status receipts

The *PostalOne!* system allows Batch Processor users to receive `Mail.dat` transaction confirmations in the form of Status Receipt files. The Status Receipt file indicates the successive stages of a `Mail.dat` transaction being processed and can be transmitted in two formats: XML and character delimited.

The following is the XML schema for Status Receipts:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xs:element name="receipts">
    <xs:complexType>
      <xs:all>
        <xs:element name="record-type">
          <xs:annotation>
            <xs:documentation>The current record type for PostalOne! receipt
files is: transfer-status. The transfer-status record shall serve to indicate the
progress of the processing of a Mail.dat file transferred to the PostalOne! Java
upload server. The record shall trace, through its fields, the validation, the
loading, and the processing of postage statements for a Mail.dat file
transfer.</xs:documentation>
          </xs:annotation>
        </xs:complexType>
      </xs:all>
    </xs:element name="version">
```

```

        <xs:annotation>
            <xs:documentation>the version of this receipt file.
(e.g. 1.0)</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="job-id">
        <xs:annotation>
            <xs:documentation>Mail.dat Job ID as presented in the
header file specification of Mail.dat.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="filename">
        <xs:annotation>
            <xs:documentation>the part of the name of a Mail.dat
file without the extension. For example, if the files sent are foo.hdr, foo.csm and
foo.cqt, the filename is foo. If the receipt record is for a finalized, a canceled or
a reversed postage statement, this record will contain the container ID of one of the
containers that was finalized, canceled, or reversed.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="verification-facility-zip-4">
        <xs:annotation>
            <xs:documentation>The zip plus 4 code of the mailing
facility where verification occurred.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="date-time">
        <xs:annotation>
            <xs:documentation>a time stamp, represented in the 24
hour time system, that indicates the creation or the receipt of a file. The Java
upload server provides the time that is used.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="validation">
        <xs:annotation>
            <xs:documentation>a binary value of P or F indicating
successful validation or failed validation, respectively, of a Mail.dat
file.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="job-acceptance">
        <xs:annotation>
            <xs:documentation>a binary value of P or F indicating
whether a file has been successfully loaded to the Java upload server's data
repository for loading to a database.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="insert">
        <xs:annotation>
            <xs:documentation>a binary value of P or F indicating
the successful loading of a job from the Java upload server's data repository to the
database.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="PS-generated">
        <xs:annotation>
            <xs:documentation>a Boolean value of T or F indicating
whether a postage statement has been generated or has not been
generated.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="PS-canceled">
        <xs:annotation>
            <xs:documentation>a value of C or null indicating
whether a postage statement has been canceled or no action has been taken,
respectively.</xs:documentation>
        </xs:annotation>
    </xs:element>

```

```

        </xs:element>
        <xs:element name="PS-finalized">
          <xs:annotation>
            <xs:documentation>a value of F, R or null indicating
whether a postage statement has been finalized, reversed or no action has been taken,
respectively.</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="mpu-edit">
          <xs:annotation>
            <xs:documentation>a value of T or null indicating
whether an mpu edit has occurred on the front-end or not.</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="error-code">
          <xs:annotation>
            <xs:documentation>a numeric error code generated by the
PostalOne! Java upload server that represents the occurrence or absence of an error. A
zero in the field indicates the absence of error.</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="error-message">
          <xs:annotation>
            <xs:documentation>a message generated by the PostalOne!
Java upload server that describes, with brevity, an error or warning that has
occurred. If there is no error or warning, this field will be null.</xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:all>
</xs:complexType>
</xs:element>
</xs:schema>

```

## Record Type: transfer-status

The current record type for *PostalOne!* receipt files is: *transfer-status*.

The *transfer-status* record shall serve to indicate the progress of the processing of a Mail.dat file transferred to the *PostalOne!* Java upload server. The record shall trace, through its fields, the validation, the loading, and the processing of postage statements for a Mail.dat file transfer. The format for job status records is as follows:

```
transfer-status<char-del>version<char-del>jobid<char-del>filename<char-del>verification-facility-
zip+4<char-del>date-time<char-del>validation<char-del>job-acceptance<char-del>insert<char-del>PS-
gen<char-del>PS-canceled<char-del>PS-fin<char-del>mpu-edit<char-del> error-code<char-del>error-
msg<new-line>
```

where

**transfer-status** : represents the literal text that will be placed at the beginning of a transfer-status record.

**<char-del>**: represents the ASCII character used as the field delimiter.

**version**: the version of this receipt file. (e.g. 1.0)

**jobid**: the Mail.dat Job ID as presented in the header file specification of Mail.dat.

**filename**: the part of the name of a Mail.dat file without the extension. For example, if the files sent are foo.hdr, foo.csm and foo.cqt, the "filename" is foo. If the receipt record is for a finalized, a canceled or a

reversed postage statement, this record will contain the container ID of one of the containers that was finalized, canceled, or reversed.

**verification-facility-zip+4:** The ZIP+4 Code of the mailing facility where verification occurred.

**date-time:** a time stamp, represented in the 24 hour time system, that indicates the creation or the receipt of a file. The Java upload server provides the time that is used.

**validation:** a binary value of "P" or "F" indicating successful validation or failed validation, respectively, of a Mail.dat file.

**job-acceptance:** a binary value of "P" or "F" indicating whether a file has been successfully loaded to the Java upload server's data repository for loading to a database.

**insert:** a binary value of "P" or "F" indicating the successful loading of a job from the Java upload server's data repository to the database.

**PS-gen:** a Boolean value of "T" or "F" indicating whether a postage statement has been generated or has not been generated.

**PS-canceled:** a value of "C" or <empty> indicating whether a postage statement has been canceled or no action has been taken, respectively.

**PS-fin:** a value of "F", "R" or <empty> indicating whether a postage statement has been finalized, reversed or no action has been taken, respectively.

**mpu-edit:** a value of "T" or <empty> indicating whether an mpu edit has occurred on the front-end or not.

**error-code:** a numeric error code generated by the *PostalOne!* Java upload server that represents the occurrence or absence of an error. A zero in the field indicates the absence of error.

**error-msg:** a message generated by the *PostalOne!* Java upload server that describes, with brevity, an error or warning that has occurred. If there is no error or warning, this field will be empty.

**<new-line>:** the platform specific new line character sequence.

## Appendix C. Postage Statement Receipt File Layout

This appendix contains details about the contents of a postage statement receipt file and the associated record types and format.

### C-1 Postage Statement Receipt File Content

The postage statement receipt file conforms to these specifications in this section. The postage statement data for the 3602-N and the 3602-R can be divided into three main sections:

- Address information – summary record
- Mailing information – summary record
- Postage Computation information – detailed calculations

The postage statement receipt data for a single postage statement shall consist of rows of character delimited data in the format summarized in the following table:

<b>Summary Record</b>	Receipt Header	Job Information	Address Information	Mailer Information
<b>Computation Records</b>	Receipt Header	Job Information	Postage Computation Information (detailed calculations)	

For a postage statement, there will be only one Summary Record but possibly several Computation Records. The Computation Records are linked to the Summary Record by the Receipt Job Information field. The table above depicts the ordering of data within a row of postage statement data for both record types. The Summary Record and the Computation records are described in detail below.

### Summary Record

The summary record is composed of a total of 136 atomic, character delimited fields of data. The details for the following summary records follow:

- Receipt Header Information
- Job Information
- Address Information
- Mailing information

**NOTE:** All fields depicted in the tables to follow are separated by the delimiter character that is specified in the “Delimiter” parameter of the `postal1.ini` file on the client machine that runs the batch processor and makes the request for the receipt.

#### Receipt Header Information (Summary Record)

Field Position	Name	Description
1	Receipt Type	The text "postage-statement"
2	Version	The version number of the format: <b>v.r</b> (where <b>v</b> is the version and <b>r</b> is the revision number) for example, 1.0
3	Date	A system generated date and in the format <b>yyyy-mm-dd</b>
4	Postage Statement Type	The postage statement type, e.g., 3602-R

### Job Information (Summary Record)

Field Position	Name	Description
5	Job ID	The Mail.dat Job ID for this job
6	File Number	A number that uniquely identifies the postage statement

### Address Information (Summary Record)

The address information consists of the Permit holder address (fields 7-24) and the Company or mailer address (fields 25-34).

Permit holder address information

Field Position	Name	Description
7	Role	The role of this addressee in this particular mailing.
8	City	City
9	Contact Name	Contact's name
10	Dun & Bradstreet No.	The Dun & Bradstreet Number (not currently populated—may be in the future)
11	E-mail Address	The permit holder's e-mail address
12	Name	Name of addressee
13	Permit Publication Number	The number of the permit or publication
14	Permit Publication Type	The type of permit or publication: PI – Permit Imprint MT – Metered Tape PC – Precanceled Stamps BR – Business Reply PE – Periodicals GH – Ghost
15	Permit_zip	Permit ZIP Code
16	Publication Title	Title of publication
17	Telephone	The permit holder's telephone number
18	Phone Extension	The permit holder's telephone extension
19	State	State
20	Street Address 1	Street address
21	Street Address 2	PO Box
22	Zip Plus 5	5-digit ZIP Code
23	Zip Plus 4	4-digit extended ZIP Code
24	Finance Number	Finance number

Mailer address information

Field Position	Name	Description
25	Name	Name of addressee
26	Address	Street address
27	City	City
28	Country Code	Postal country code
29	State	State
30	Zip Plus 5	5-digit ZIP Code
31	Zip Plus 4	4-digit extended ZIP Code
32	Telephone	The mailing agent's telephone number
33	Dun & Bradstreet No.	The Dun & Bradstreet Number (not currently populated—may be in the future)
34	Permit_zip	The zip of the permit or publication

Mailing information (Summary Record)

Field Position	Name	Description
35	Statement Sequence Number	Postage Statement Sequence Number uniquely associated with a postage statement
36	Form name	Name of the form
37	Form Type	Type of statement being sent: PE – Periodicals PM – Priority Mail FC – First-Class Mail SM – Standard Mail BP – Bound Printed Matter PP – Parcel Post ML – Media / Library Mail
38	Stage	Current stage of the statement: UPD – indicates an “Actual” statement that has been submitted by the user but not verified. FIN – indicates a “Finalized” statement that has been verified. REV – indicates a statement that was previously finalized but is now reversed. CAN – indicates a statement that was never finalized and is currently canceled and unavailable for further modification.
39	Permit Publication Number	The number of the permit or publication
40	Permit Publication Type	The type of permit or publication: PI – Permit Imprint MT – Metered Tape PC – Precanceled Stamps BR – Business Reply PE – Periodicals GH – Ghost

Field Position	Name	Description
41	Rate Type	The special rate that applies to the mail sent: R – Regular N – Nonprofit C – Classroom S – Science of Agriculture T – Priority Mail B – Bound Printed Matter P – Parcel Post L – Library Mail F – Media Mail Periodicals: C – Classroom N – Nonprofit R – Regular S – Science of Agriculture Priority Mail: T – Priority Mail First Class: R – Regular <u>Standard Mail:</u> R – Regular N – Nonprofit <u>Parcel Post:</u> P – Parcel Post <u>Bound Printed Matter:</u> B – Bound Printed Matter <u>Media Library Mail:</u> L – Library F – Media Mail
42	Processing Category	L – Letters F – Flats A – Automation Flats
43	Class	Mail class of the piece: 1 – First Class 2 – Periodicals 3 – Standard Mail 4 – Package Services
44	Mailing Date	Date on which the pieces are delivered to the Post Office. <b>NOTE:</b> This is the date applicable to the postage rates.
45	Weight of a Single Piece	Weight of a single piece if the mailing is an identical-weight mailing
46	Special Fee	Total dollar amount of any special fees from a form 3540-S associated with this mailing. [Should be system generated when we build support for the 3540-S]
47	Total Pieces	Total pieces in mailing
48	Total Weight	Total weight of mailing
49	Total Postage	Total postage of the mailing as calculated by the system

Field Position	Name	Description
50	Automation Rate Pieces, Date of Address Matching and Coding	For Automation Rate Pieces, Date of Address Matching and Coding (DMM A950.3.0) (either the date in the format <b>yyyy-mm-dd</b> or the text "Not Available")
51	Enhanced Carrier Route Rate Pieces, Enter Date of Address Matching and Coding	Enhanced Carrier Route Rate Pieces, Enter Date of Address Matching and Coding (DMM A050.4.0) (either the date in the format <b>yyyy-mm-dd</b> or the text "Not Available")
52	CR_SEQ_DATE	CR sequencing date in M_POSTAGE_STATEMENT_SUM
53	Post Office Zip	Mailing facility
54	Activity Date	Date of last update
55	Finance Number	Finance number
56	Job Id	The Mail.dat Job ID for this job
57	JOB_PROVIDER_CODE	The Mail.dat provider code for this job
58	MAILING_FACILITY	The Mail.dat mailing facility for this job.
59	JOB_IDEA_VER	Version of the job
60	Origin	Job origin
61	Pallets	Number of pallets
62	Sacks	Number of sacks
63	Flat Trays	Number of flat trays
64	2'EMM Trays	Number of 2'EMM Trays
65	1'MM Trays	Number of 1'MM Trays (data for the number of containers starts here)
66	2' MM Trays	Number of 2' MM Trays
67	Other	Number of other units (data for the number of containers ends here)
68	total_cost_parta	Total cost of part a
69	total_cost_partb	Total cost of part b
70	Total_cost_partc	Total cost of part c
71	Total_cost_partd	Total cost of part d

## Computation Record

The computation records for the Mail.dat receipt indicate details of the postage statement calculation. While there is only one summary record, there may be one or more computation records belonging to a receipt. A single computation record is composed of 33 atomic fields of data. The details for the following computation records follow:

- Receipt Header Information
- Job Information
- Postage Computation Information—detailed calculations

### Receipt Header Information (Computation Record)

Field Position	Name	Description
1	Receipt Type	The text "postage-statement"
2	Version	The version number of the format: <b>v.r</b> (where <b>v</b> is the version and <b>r</b> is the revision number) for example, 1.0
3	Date	A system generated date and in the format <b>yyyy-mm-dd</b>
4	Postage Statement Type	The postage statement type e.g., 3602-R

### Job Information (Computation Record)

Field Position	Name	Description
5	Job ID	The Mail.dat Job ID for this job
6	File Number	A number that uniquely identifies the postage statement

### Postage Computation Information (Detailed Calculations)

Field Position	Name	Description
7	Line Number	Line Number
8	Line Label	English name of the line on which the pieces would appear
9	Piece Rate	Rate applied per piece to the pieces with these characteristics
10	Unit Weight	Weight of a single piece on this line, in pounds, to 4 decimal places (for example, 1.2345) <ul style="list-style-type: none"><li>• Periodicals: 70 lbs (1120 oz)</li><li>• Priority Mail: 70 lbs (1120 oz)</li><li>• First Class Mail:<ul style="list-style-type: none"><li>• Automation Letters: 0.2187 lbs (3.5 oz)</li><li>• Automation Flats: 0.8125 lbs (13 oz)</li><li>• Non-automation: 0.8125 lbs (13 oz)</li></ul></li><li>• Standard Mail: 0.99999999 lbs (16 oz)</li><li>• Parcel Post 15 lbs (240 oz)</li><li>• Bound Printed: Matter: 15 lbs (240 oz)</li><li>• Media / Library Mail: 70 lbs (1120 oz)</li></ul>
11	Piece Postage	Total postage for the pieces on this line
12	Pound Postage	Total postage for the pounds on this line

## C-2 Record Types and Record Format

This section provides details about the postage statement file's record type and format. For additional information about postal1.ini parameters, see the table in Sections 4.1.4, Downloading and Configuring the NT Batch Processor or 4.1.5, Downloading and Configuring the UNIX Batch Processor.

### Record Type: postage-statement

The *PostalOne!* system allows only batch processor users to receive Mail.dat transaction statements in the form of postage statement receipt files. The postage statement receipt file reports the statement calculation and the information used to arrive at it. It can be transmitted in two formats: XML and character delimited.

## The following is the XML schema for Postage Statement Receipts:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="com.postagestatementwizard.data"
xmlns="com.postagestatementwizard.data" xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="PostageStatementData">
    <xs:annotation>
      <xs:documentation>Generic Postage Statement Form data (ROOT)</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="postageStatementID" type="xs:string" nillable="false">
          <xs:annotation>
            <xs:documentation>USPS primary key</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="formType" nillable="false">
          <xs:annotation>
            <xs:documentation>Processing Form Type</xs:documentation>
          </xs:annotation>
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:maxLength value="2"/>
              <xs:enumeration value="SM"/>
              <xs:enumeration value="FC"/>
              <xs:enumeration value="PM"/>
              <xs:enumeration value="BP"/>
              <xs:enumeration value="PP"/>
              <xs:enumeration value="ML"/>
              <xs:enumeration value="PE"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element name="formNumber">
          <xs:annotation>
            <xs:documentation>USPS form number</xs:documentation>
          </xs:annotation>
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:maxLength value="20"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element ref="MailerData" maxOccurs="4"/>
        <xs:element ref="MailingData"/>
        <xs:element ref="CertificationData"/>
        <xs:element ref="ContainerData" maxOccurs="unbounded"/>
        <xs:element ref="PeriodicalData" minOccurs="0"/>
        <xs:element ref="LineItemData" maxOccurs="unbounded"/>
        <xs:element ref="MailDat" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Phone">
    <xs:annotation>
      <xs:documentation>Global phone definition</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="areaCode" nillable="false">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:minLength value="3"/>
              <xs:maxLength value="3"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```

</xs:element>
<xs:element name="prefix" nillable="false">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:minLength value="3"/>
      <xs:maxLength value="3"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="exchange" nillable="false">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:minLength value="4"/>
      <xs:maxLength value="4"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="extension" nillable="true">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="5"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="Address">
  <xs:annotation>
    <xs:documentation>Global address definition</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="address1" nillable="false">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="40"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="address2" nillable="true">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="40"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="city" nillable="true">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="40"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="state" nillable="true">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="2"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="zip" type="ZIP" nillable="true"/>
      <xs:element name="zip4" nillable="true">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="4"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

```

        </xs:simpleType>
    </xs:element>
    <xs:element name="country_code" nillable="true">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:maxLength value="2"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="MailerData" nillable="true">
    <xs:annotation>
        <xs:documentation>Global MailerData</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:sequence>
            <xs:element name="mailerRole" nillable="false">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:minLength value="2"/>
                        <xs:maxLength value="2"/>
                        <xs:enumeration value="PU"/>
                        <xs:enumeration value="MO"/>
                        <xs:enumeration value="MA"/>
                        <xs:enumeration value="PH"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="name" type="xs:string" nillable="false"/>
            <xs:element ref="Phone"/>
            <xs:element ref="Address"/>
            <xs:element name="emailAddress" nillable="true">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:maxLength value="40"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="capsID" nillable="true">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:maxLength value="20"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="dunsNumber" nillable="true">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:maxLength value="20"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="permitNumber" nillable="true">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:maxLength value="20"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="permitType" nillable="true">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:maxLength value="2"/>
                        <xs:enumeration value="PI"/>
                        <xs:enumeration value="MT"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>

```

```

        <xs:enumeration value="PE"/>
        <xs:enumeration value="GH"/>
        <xs:enumeration value="BR"/>
        <xs:enumeration value="PC"/>
    </xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="permitZip" nillable="true">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:maxLength value="5"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
<xs:element name="contactPerson" minOccurs="0">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:maxLength value="25"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
<xs:element name="financeNumber" minOccurs="0">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:maxLength value="6"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="CertificationData">
    <xs:annotation>
        <xs:documentation>Global CertificationData</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:sequence>
            <xs:element name="name" nillable="false">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:maxLength value="40"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element ref="Phone"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="ContainerData">
    <xs:annotation>
        <xs:documentation>Global ContainerData</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:sequence>
            <xs:element name="containerType" nillable="false">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:enumeration value="tray1mm"/>
                        <xs:enumeration value="tray2mm"/>
                        <xs:enumeration value="tray2emm"/>
                        <xs:enumeration value="flat"/>
                        <xs:enumeration value="sack"/>
                        <xs:enumeration value="pallet"/>
                        <xs:enumeration value="other"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>

```

```

        <xs:element name="containerNumber" type="xs:long" nillable="false"/>
    </xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="PeriodicalData">
    <xs:annotation>
        <xs:documentation>Global PeriodicalData</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:sequence>
            <xs:element name="publicationTitle">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:maxLength value="60"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="over10" type="xs:boolean"/>
            <xs:element name="rideAlong" type="xs:boolean"/>
            <xs:element name="issueDate" type="xs:date"/>
            <xs:element name="issueFrequency" type="xs:string"/>
            <xs:element name="editionCode" type="xs:string"/>
            <xs:element name="mailingType" type="xs:string"/>
            <xs:element name="numberIssues" type="xs:long"/>
            <xs:element name="advertisingPercent" type="xs:decimal"/>
            <xs:element name="sheetWeight" type="xs:decimal"/>
            <xs:element name="copyWeight" type="xs:decimal"/>
            <xs:element name="rideAlongWeight" type="xs:decimal"/>
            <xs:element name="totalCopies" type="xs:long"/>
            <xs:element name="statementType">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:maxLength value="1"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="recipientType">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:maxLength value="1"/>
                        <xs:enumeration value="0"/>
                        <xs:enumeration value="1"/>
                        <xs:enumeration value="2"/>
                        <xs:enumeration value="3"/>
                        <xs:enumeration value="4"/>
                        <xs:enumeration value="5"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="LineItemData">
    <xs:annotation>
        <xs:documentation>Global LineItemData</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:sequence>
            <xs:element name="section" nillable="false">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:maxLength value="1"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="lineNumber" minOccurs="0">
                <xs:simpleType>

```

```

        <xs:restriction base="xs:string">
            <xs:maxLength value="3"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
<xs:element name="lineLabel" minOccurs="0">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:maxLength value="80"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
<xs:element name="rateCategory" nillable="false">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:maxLength value="2"/>
            <xs:enumeration value="A"/>
            <xs:enumeration value="B"/>
            <xs:enumeration value="D"/>
            <xs:enumeration value="E"/>
            <xs:enumeration value="F"/>
            <xs:enumeration value="G"/>
            <xs:enumeration value="H"/>
            <xs:enumeration value="J"/>
            <xs:enumeration value="K"/>
            <xs:enumeration value="L"/>
            <xs:enumeration value="L1"/>
            <xs:enumeration value="L2"/>
            <xs:enumeration value="L3"/>
            <xs:enumeration value="L4"/>
            <xs:enumeration value="N"/>
            <xs:enumeration value="O"/>
            <xs:enumeration value="P"/>
            <xs:enumeration value="Q"/>
            <xs:enumeration value="S"/>
            <xs:enumeration value="X"/>
            <xs:enumeration value="Y"/>
            <xs:enumeration value="Z"/>
            <xs:enumeration value="Z1"/>
            <xs:enumeration value="Z2"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
<xs:element name="singlePieceWeight" type="xs:decimal"/>
<xs:element name="totalPieces" type="xs:long" nillable="false"/>
<xs:element name="totalWeight" type="xs:decimal" nillable="true"/>
<xs:element name="entryDiscount" nillable="false">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:maxLength value="1"/>
            <xs:enumeration value="N"/>
            <xs:enumeration value="B"/>
            <xs:enumeration value="S"/>
            <xs:enumeration value="D"/>
            <xs:enumeration value="A"/>
            <xs:enumeration value="P"/>
            <xs:enumeration value="Q"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
<xs:element name="barcodeRate" nillable="false">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:maxLength value="1"/>
            <xs:enumeration value="Y"/>
            <xs:enumeration value="N"/>
            <xs:enumeration value="O"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>

```

```

        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="surchargeType" nillable="false">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:maxLength value="1"/>
          <xs:enumeration value="N"/>
          <xs:enumeration value="R"/>
          <xs:enumeration value="Q"/>
          <xs:enumeration value="O"/>
          <xs:enumeration value="S"/>
          <xs:enumeration value="2"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="zone" nillable="false">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:maxLength value="2"/>
          <xs:enumeration value="N"/>
          <xs:enumeration value="LC"/>
          <xs:enumeration value="1"/>
          <xs:enumeration value="2"/>
          <xs:enumeration value="3"/>
          <xs:enumeration value="4"/>
          <xs:enumeration value="5"/>
          <xs:enumeration value="6"/>
          <xs:enumeration value="7"/>
          <xs:enumeration value="8"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="pieceRate" type="xs:decimal"/>
    <xs:element name="poundRate" type="xs:decimal"/>
    <xs:element name="postageAmount" type="xs:decimal"/>
    <xs:element name="preparedAsSM" type="xs:boolean"/>
    <xs:element name="heavyLetterIndicator" type="xs:boolean"/>
    <xs:element name="entryFacilityType">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:maxLength value="1"/>
          <xs:enumeration value="N"/>
          <xs:enumeration value="B"/>
          <xs:enumeration value="S"/>
          <xs:enumeration value="D"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="entryZip" type="ZIP"/>
  </xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="MailingData">
  <xs:annotation>
    <xs:documentation>Global MailingData</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="mailingDate" type="xs:date" nillable="false"/>
      <xs:element name="postOfficeOfMailingZip" type="ZIP"/>
      <xs:element name="postagePaymentMethod" nillable="false">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="1"/>
            <xs:enumeration value="P"/>
            <xs:enumeration value="S"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

```

        <xs:enumeration value="M"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="processingCategory" nillable="false">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="2"/>
        <xs:enumeration value="LT"/>
        <xs:enumeration value="FL"/>
        <xs:enumeration value="CD"/>
        <xs:enumeration value="MP"/>
        <xs:enumeration value="IR"/>
        <xs:enumeration value="PF"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="rateType" nillable="false">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="1"/>
        <xs:enumeration value="R"/>
        <xs:enumeration value="N"/>
        <xs:enumeration value="C"/>
        <xs:enumeration value="S"/>
        <xs:enumeration value="T"/>
        <xs:enumeration value="B"/>
        <xs:enumeration value="P"/>
        <xs:enumeration value="L"/>
        <xs:enumeration value="F"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="singlePieceWeight" type="xs:decimal"/>
  <xs:element name="federalAgencyCostCode">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="20"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="statementSequenceNumber">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="20"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="sackingBasis" nillable="true">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="1"/>
        <xs:enumeration value="0"/>
        <xs:enumeration value="1"/>
        <xs:enumeration value="2"/>
        <xs:enumeration value="3"/>
        <xs:enumeration value="4"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="packagingMethod" nillable="true">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="1"/>
        <xs:enumeration value="0"/>
        <xs:enumeration value="1"/>
        <xs:enumeration value="2"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>

```

```

        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="enclosedBy" nillable="true">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:maxLength value="2"/>
          <xs:enumeration value="SM"/>
          <xs:enumeration value="PE"/>
          <xs:enumeration value="BP"/>
          <xs:enumeration value="PP"/>
          <xs:enumeration value="ML"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="automationAddressMatchDate" type="xs:date"/>
    <xs:element name="carrierRouteSequencingDate" type="xs:date"/>
    <xs:element name="carrierRouteAddressMatchDate" type="xs:date"/>
    <xs:element name="postageAffixedPieces" type="xs:long"/>
    <xs:element name="postageAffixedAmount" type="xs:decimal"/>
    <xs:element name="specialFeesAmount" type="xs:decimal"/>
    <xs:element name="registerOfMailing" type="xs:boolean"/>
    <xs:element name="experimentalIndicator" type="xs:boolean"/>
    <xs:element name="totalPieces" type="xs:long"/>
    <xs:element name="totalWeight" type="xs:decimal"/>
    <xs:element name="totalPostage" type="xs:decimal"/>
    <xs:element name="federalAgencyCode">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:maxLength value="20"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="internationalInboundIndicator" type="xs:boolean"/>
  </xs:sequence>
</xs:complexType>
</xs:element>
<xs:simpleType name="ZIP">
  <xs:annotation>
    <xs:documentation>Global ZIP type</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:maxLength value="5"/>
  </xs:restriction>
</xs:simpleType>
<xs:element name="MailDat">
  <xs:annotation>
    <xs:documentation>Global MailDat data</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="jobID" type="xs:string"/>
      <xs:element name="providerCode" type="xs:string"/>
      <xs:element name="mailingFacility" type="xs:string"/>
      <xs:element name="ideaVersion" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>

```

## Appendix D. Validation Log File Error Messages

This appendix contains a complete listing of error messages that are written to the Validation.log file when the File Validator encounters errors. By default, the file is on your machine in C:\Validation.log.

**Table D-1. Validation Log File Error Messages**

Error Message	Reason	Explanation
[Job ID] – [File Name] - [Date] [Time] – ERROR - In "file name" mandatory field "field name" (Row #: Position [start] #_[stop] #) is blank.	Mandatory field is blank.	A mandatory field in the reported file contains no data. In the error message, Row # refers to the record number within the failed file that is missing data. Position [start] #_[stop] # refers to the position (i.e., field) within the row that is missing data.
[Job ID] – [File Name] - [Date] [Time] – ERROR - In "file name" mandatory field "field name" (Row #: Position [start] #_[stop] #) contains invalid data.	Mandatory field contains invalid data.	A mandatory field in the reported file contains invalid data. In the error message, Row # refers to the record number within the failed file that is missing data. Position [start] #_[stop] # refers to the position (i.e., field) within the row that contains invalid data.
[Job ID] – [File Name] - [Date] [Time] – ERROR - In "file name" optional field "field name" (Row #: Position [start] #_[stop] #) contains invalid data.	Optional field contains invalid data.	An optional field in the reported file contains invalid data. In the error message, Row # refers to the record number within the failed file that is missing data. Position [start] #_[stop] # refers to the position (i.e., field) within the row that contains invalid data.
[Job ID] – [File Name] - [Date] [Time] – ERROR - In "file name" the "field name" (Row #: Position [start] #_[stop] #) and "field name" (Row #: Position [start] #_[stop] #) fields contain incompatible data.	Two fields contain incompatible data.	The fields in the reported file contain incompatible data. This occurs when reported information between related fields is incompatible. In the error message, Row # refers to the record numbers within the failed files that are incompatible.
[Job ID] – [File Name] - [Date] [Time] – ERROR - In "file name" field "field name" (Row #: Position [start] #_[stop] #) should be mutually exclusive to the "USPS Pub Number" (Row #: Position [start] #_[stop] #) field, but both fields contain data.	Two fields contain data where only one should.	The referenced fields both contain data. In this type of mailing, one or the other field may contain data, but not both. In the error message, Row # refers to the record numbers within the failed file that contain data. You must remove data from one of the fields to submit this file.
[Job ID] – [File Name] - [Date] [Time] – ERROR - In "file name" either the "field name" (Row #: Position [start] #_[stop] #) field or the "field name" (Row #: Position [start] #_[stop] #) field should be filled, but both fields are blank.	Two fields; at least one of which should contain data, but both are blank.	Neither field referenced contains data. In this mailing, one of these fields must contain data, but both are blank. In the error message, Row # refers to the record numbers within the failed files that are missing data. You must supply the data for one of these fields to submit this file.
[Job ID] – [File Name] - [Date] [Time] – ERROR - In "file name" length of "record #" is incorrect.	Incorrect Record Length	A record of incorrect length was found in one of the Mail.dat files.

Error Message	Reason	Explanation
<p>ABCD1234 - ABCD1234.csm - 01-Aug-01 6:10 PM – ERROR - In "c:\maildat\ABCD1234.csm" Row #2, the CONTAINER_ID and the SIBLING_CONTAINER_REF_ID fields cannot contain the same values.</p>	<p>Incorrect Reference</p>	<p>A container that references itself.</p>
<p>ABCD1234 - ABCD1234.mpu - 01-Aug-01 6:19 PM – ERROR - In "c:\maildat\ABCD1234.mpu" The CO_PALLETIZATION_CODE field in Record #1 cannot be blank, because the corresponding WSR record is present.</p>	<p>Fields that should be related are not.</p>	<p>If the Walk Sequence (.wsr) file is used in the Mail.dat transfer, the Co-Palletization Code field in the Mailpiece Unit (.mpu) record must be present.</p>
<p>ABCD1234 - ABCD1234.csm - 27-Jun-01 3:33 PM – ERROR - In "c:\maildat\ABCD1234.csm" The Key fields of record #5 are duplicated within this file.</p>	<p>Duplicate Key</p>	<p>The key fields in a record are found to be the same in another record.</p>
<p>ABCD1234 - ABCD1234.cqt - 01-Aug-01 7:02 PM – ERROR - In "c:\maildat\ABCD1234.cqt" the corresponding Key fields of record #766 do not match with a CSM record in this job.</p>	<p>Invalid Key</p>	<p>The key fields in a child record do not correspond to a parent record.</p>
<p>ABCD1234 - ABCD1234.mpu - 01-Aug-01 7:02 PM – ERROR - In file "c:\maildat\ABCD1234.mpu" Record #3, the RATE_CATEGORY field is set to 'A', but no corresponding WSR record is present.</p>	<p>Field level information indicates that another file should be present, but it cannot be found.</p>	<p>A rate category claims saturation walk sequence; as a result, we would expect to see a Walk Sequence Record.</p>
<p>ABCD1234 - ABCD1234.csm - 01-Aug-01 6:16 PM – ERROR - The 'CSM' record count '49' as indicated in the "c:\maildat\ABCD1234.hdr" file do not match the actual number of record(s) '51' found in the "c:\maildat\ABCD1234.csm" file.</p>	<p>Count Mismatch</p>	<p>The Header record count does not match the actual record count.</p>
<p>ABCD1234 - ABCD1234.csm - 27-Jun-01 3:44 PM – ERROR - In "c:\maildat\ABCD1234.csm" The following Sibling Container Reference ID(s) do not correspond to any original CSM record(s) in this file. Sibling Container Ref ID: 000101 Sibling Container Ref ID: 000100</p>	<p>Reference to a non-existing record.</p>	<p>Reference is made to a file that does not exist.</p>
<p>ABCD1234 - ABCD1234.mcr - 02-Aug-01 1:12 PM - ERROR - In "c:\maildat\ABCD1234.mcr" the corresponding key fields of the following MCR record do not match with any MPU record(s) in this job. JOB_ID: 'ABCD1234', COMPONENT_ID: 'S3S3 ', SEGMENT_ID: 'S3 ', MPU_ID: 'S4 ' The following MPU record(s) do not contain any corresponding MCR record(s): Job_id: '00004230', Segment_id: 'S3 ', MPU_id: 'S3 '</p>	<p>Key Violation</p>	<p>In this case, there could be error messages for all corresponding files in a relationship: Mailpiece Unit, Component, and the MPU/C Relationship records.</p>

Error Message	Reason	Explanation
ABCD1234 – ABCD1234.mcr - 02-Aug-01 4:20 PM – ERROR - There must be at least one or more MCR record(s) present in file "c:\abcd1234.mcr" for every MPU record found in file "c:\abcd1234.mpu".The following MPU record(s) do not contain any corresponding MCR record(s): Job_id: '00000001', Segment_id: 'S1 ', MPU_id: '001'	Missing a required file.	The relationship between the mailpiece unit records and the component records must be defined with the mpu/c relationship record.
ABCD1234 – abcd1234.seg - 07-Mar-03 15:54 PM - ERROR - The last record in file "C:\abcd1234.seg" does not end with a newline character.	Missing a 'newline' character.	All Mail.dat records must end with a 'newline' (enter) character.
N/A – abcd1234.hdr - 07-Mar-03 15:53 PM - Error - In file "C:\abcd1234.hdr" the IDEAlliance version field is missing.	Missing Mail.dat version.	The IDEAlliance field in the Mail.dat header file is missing.
N/A – abcd1234.hdr - 07-Mar-03 15:52 PM - Error - In "C:\abcd1234.hdr" . The Mail.dat file version "03-1" is not currently supported.	Unsupported Mail.dat version.	The <i>PostalOne!</i> system does not currently support the Mail.dat file version specified in the error message.
ABCD1234 – abcd1234.seg - 03-Mar-03 08:16 AM - ERROR - The record status value specified for record #1 in file "C:\abcd1234.seg" does not match with the file level status specified in the header file.	Incompatible File/Record level status flags.	See Section 3.3.3.1 for file/record level status flag usage requirements.
ABCD1234 – abcd1234.hdr - 25-Feb-03 14:26 PM - ERROR - The 'C:\abcd1234.hdr' file contains duplicate history sequence numbers .	Duplicate 'history' sequence numbers.	The 'history' header records sequence numbers must be unique.
ABCD1234 – abcd1234.hdr - 12-Feb-03 12:01 PM - Error - In "C:\abcd1234.hdr" Error: All file-level status flags are not identical in the Header file: 'C:\abcd1234.hdr'.	Inconsistent file level status flags.	All 'file' level status flags must be consistent in a Mail.dat header file.
ABCD1234 – abcd1234.hdr - 21-Feb-03 18:16 PM - Error - In "C:\abcd1234.hdr" Mail.dat transmittal for JOB ID ABCD1234 attempt to sent without a 'C:\abcd1234.SEG' file.	Missing a required file.	A required Mail.dat file is missing from this transaction.
N/A – abcd1234.hdr - 07-Mar-03 15:52 PM - ERROR - In File "C:\ abcd1234.hdr", Job ID is either missing or not valid.	Missing or Invalid Job ID.	The Job ID field must be populated.
ABCD1234 - abcd1234.hdr - 07-Mar-03 15:52 PM - ERROR - In File "C:\ abcd1234.hdr", Provider code is either missing or not valid.	Missing Provider Code.	The Provider Code field must be populated.
ABCD1234 - abcd1234.hdr - 07-Mar-03 15:52 PM - ERROR - In File "C:\ abcd1234.hdr" file contains an Invalid history status.	Invalid history status.	The header history status field must contain a 'C' or 'H'.

Error Message	Reason	Explanation
ABCD1234 - abcd1234.hdr - 07-Mar-03 15:52 PM - ERROR - In File "C:\ abcd1234.hdr", History SEQ Number is either missing or not valid.	Missing or Invalid history sequence number.	The header history sequence number must be populated.
ABCD1234 - abcd1234.hdr - 07-Mar-03 15:52 PM - ERROR - In File "C:\ abcd1234.hdr", one or more Record Counts are either missing or not valid.	Missing or invalid record counts.	All record count fields in the header file must be populated.
N/A - abcd1234.hdr - 07-Mar-03 15:52 PM - ERROR - In File "C:\ abcd1234.hdr", one or more Record Closing Characters are either missing or not valid.	Missing or invalid closing characters.	All Mail.dat records must contain a '#' closing character.
ABCD1234 - abcd1234.hdr - 07-Mar-03 15:52 PM - ERROR - The header file contains multiple current status lines in the History Status field.	Multiple 'current' header records.	A Mail.dat job must contain only one 'current' header record.
ABCD1234 - abcd1234.hdr - 07-Mar-03 15:52 PM - ERROR - The header file contains no current status lines in the History Status field.	No 'current' header records.	A Mail.dat job must contain one 'current' header record.
ABCD1234 - abcd1234.hdr - 07-Mar-03 15:52 PM - ERROR - In File "C:\ abcd1234.hdr" file contains an Invalid Mail.dat Presentation Category.	Invalid Mail.dat presentation category.	The presentation category field in the header file contains an invalid identifier.
ABCD1234 - abcd1234.hdr - 07-Mar-03 15:52 PM - ERROR - At least one file-level status flag was not accepted in the Header file: "C:\ abcd1234.hdr".	Invalid file level status flag.	The header file must contain valid status flags for all Mail.dat files.
ABCD1234 - abcd1234.hdr - 07-Mar-03 15:52 PM - ERROR - In File "C:\ abcd1234.hdr" file contains Non-zero record counts that are not allowed in the header record of a delete job (except Segment record count).	Non-zero record counts indicated.	Non-zero record counts indicated for Mail.dat files that are not allowed for a 'delete' transaction.

## Appendix E. File Transfer Error Messages

This appendix contains the error codes and error messages that are generated as a result of manual mode or batch mode file transfers.

Server#.log (# = 1 to n), General status information.

User#.log (#1 = 1 to n), Transfer error messages containing the error codes and messages as shown in this section.

**Table E-1. File Transfer Error Messages**

File Error Messages
System Error (1000): Unable to write the HDR file (*.hdr) to the <i>PostalOne!</i> server. Please resubmit the job. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.
System Error (1001): Unable to write the SEG file (*.seg) to the <i>PostalOne!</i> server. Please resubmit the job. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.
System Error (1002): Unable to write the MPU file (*.mpu) to the <i>PostalOne!</i> server. Please resubmit the job. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.
System Error (1003): Unable to write the MCR file (*.mcr) to the <i>PostalOne!</i> server. Please resubmit the job. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.
System Error (1004): Unable to write the CPT file (*.cpt) to the <i>PostalOne!</i> server. Please resubmit the job. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.
System Error (1005): Unable to write the CSM file (*.csm) to the <i>PostalOne!</i> server. Please resubmit the job. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.
System Error (1006): Unable to write the CLR file (*.clr) to the <i>PostalOne!</i> server. Please resubmit the job. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.
System Error (1007): Unable to write the ICL file (*.icl) to the <i>PostalOne!</i> server. Please resubmit the job. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.
System Error (1008): Unable to write the CQT file (*.cqt) to the <i>PostalOne!</i> server. Please resubmit the job. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.
System Error (1009): Unable to write the PQT file (*.pqt) to the <i>PostalOne!</i> server. Please resubmit the job. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.
System Error (1010): Unable to write the WSR file (*.wsr) to the <i>PostalOne!</i> server. Please resubmit the job. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.
System Error (1011): Unable to write the SNR file (*.snr) to the <i>PostalOne!</i> server. Please resubmit the job. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.
System Error (1012): Unable to write the PLR file (*.plr) to the <i>PostalOne!</i> server. Please resubmit the job. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.
System Error (1013): Unable to write the ICR file (*.icr) to the <i>PostalOne!</i> server. Please resubmit the job. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.
System Error (1014): Unable to write the PAR file (*.par) to the <i>PostalOne!</i> server. Please resubmit the job. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.
System Error (1015): Unable to write the MSR file (*.msr) to the <i>PostalOne!</i> server. Please resubmit the job. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.
System Error (1016): Unable to write the MIR file (*.mir) to the <i>PostalOne!</i> server. Please resubmit the job. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.
System Error (1017): Unable to write the SPR file (*.spr) to the <i>PostalOne!</i> server. Please resubmit the job. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.

## File Error Messages

System Error (1018): Unable to write the SFR file (\*.sfr) to the *PostalOne!* server. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

System Error (1021): Unable to create a file on the *PostalOne!* server. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

System Error (1100): A system error occurred (error on server while reading HTTP input) while processing this transaction. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center and report this error.

User Error (1311): The *PostalOne!* system does not currently support this Mail.dat version. Please check the online documentation or call the *PostalOne!* Customer Care Center for the supported Mail.dat version(s).

System Error (2001): A System Error occurred while processing this transaction. Please call the *PostalOne!* Customer Care Center and report this error.

System Error (3003): Unable to update the transaction status in the *PostalOne!* database. Please call the *PostalOne!* Customer Care Center and report this error.

System Error (3004): Unable to update the transaction status details in the *PostalOne!* database. Please call the *PostalOne!* Customer Care Center and report this error.

System Error (3005): Unable to update the file status details in the *PostalOne!* database. Please call the *PostalOne!* Customer Care Center and report this error.

User Error (3100): An attempt was made to reinsert a Mail.dat job that already exists in the *PostalOne!* database. Such transactions are not allowed by the *PostalOne!* application.

System Error (3101): An error occurred while processing a database (SQL) transaction. Please call the *PostalOne!* Customer Care Center and report this error.

User Error (3102): The mailing facility ZIP+4 used in this Mail.dat transaction is not associated with your user account. Please correct the ZIP+4 and resubmit the job. If the ZIP+4 is correct, please call the *PostalOne!* Customer Care Center.

System Error (3103): A system error occurred while allocating a data buffer in the *PostalOne!* application. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

Error (4000): An unknown protocol (other than HTTP/HTTPS) attempted to initiate communication with the *PostalOne!* server. If using the Batch Processor, please check your client.config file and resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

Error (4001): A communication error occurred (a request other than 'POST' was used), while processing this transaction. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

Error (4002): A communication error occurred (unable to determine the content type values), while processing this transaction. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

Data error (5000): One or more duplicate records were found while processing this job. Please correct the Mail.dat data and resubmit the job.

User Error (5002): Either the Permit Number or the USPS Pub Number must be populated in all Mail.dat jobs. Both fields were blank in this transaction. Please populate the missing information and resubmit the job.

System Error (5004): A system error occurred while running an internal database stored procedure for this original job. Please call the *PostalOne!* Customer Care Center and report this error.

System Error (5005): A system error occurred while running an internal database stored procedure for this update job. Please call the *PostalOne!* Customer Care Center and report this error.

## File Error Messages

User Error (5006): No original Mail.dat job was found for this update transaction. Please submit the original Mail.dat job first, and then resubmit this transaction.

System Error (5007): A system error occurred while running an internal database stored procedure (container quantity procedure). Please call the *PostalOne!* Customer Care Center and report this error.

System Error (5008): A system error occurred while inserting the Level IDs in the *PostalOne!* database. Please call the *PostalOne!* Customer Care Center and report this error.

User Error (5009): This job is closed. Please contact your local detached mail unit (DMU).

System Error (5010): A system error occurred while running an internal database stored procedure (total weight procedure). Please call the *PostalOne!* Customer Care Center and report this error.

User Error (5014): One or more containers in this shipment have previously been identified as having paid postage. Please correct the Mail.dat data and resubmit the job.

User Error (5015): Preliminary postage statements cannot be created using a container that has already generated a postage statement. Please correct the Mail.dat data and resubmit the job.

User Error (5016): The status of a canceled or paid container cannot be changed. Please correct the Mail.dat data and resubmit the job.

System Error (5017): An error occurred (too many rows) while generating the postage statement for this transaction. Please call the *PostalOne!* Customer Care Center and report this error.

System Error (5018): An error occurred (PS exception) while generating the postage statement for this transaction. Please call the *PostalOne!* Customer Care Center and report this error.

System Error (5020): An error occurred while canceling containers in this transaction. Please call the *PostalOne!* Customer Care Center and report this error.

User Error (5021): One or more containers not marked as ready to pay have been flagged for a transportation update. Please correct the Mail.dat data and resubmit the job.

User Error (5022): The status of a container that has not been marked as ready to pay cannot be changed. Please correct the Mail.dat data and resubmit the job.

System Error (6000): The *PostalOne!* database is temporarily unavailable (server cannot connect to the database). Your transaction will be automatically processed when the *PostalOne!* database is back online.

User Error (6003): This Mail.dat transaction type is currently not supported by the *PostalOne!* system. Please correct the Mail.dat data and resubmit the job.

User Error (7000): User logon failed. Please check your user name and password and resubmit the job.

System Error (8000): A database error occurred (unique limitation error) while processing this transaction. Please call the *PostalOne!* Customer Care Center and report this error.

System Error (8001): A database configuration error has occurred. Please call the *PostalOne!* Customer Care Center and report this error.

System Error (9000): The *PostalOne!* server was unable to forward this manifest file to PTS. Please call the *PostalOne!* Customer Care Center and report this error.

Data Error (ORA-00001): A duplicate CQT record was found on line number 3519, while processing file qwbn0496.cqt. Please correct the Mail.dat file and resubmit the job.

System Error (ORA-00060): A resource deadlock condition was detected while processing file s1433556.csm (line 52). Your transaction will be automatically processed when the *PostalOne!* database is back online.

System Error (ORA-00904): A system error was encountered (invalid database column name) while processing this transaction. Please call the *PostalOne!* Customer Care Center and report this error.

System Error (ORA-01033): *PostalOne!* database initialization or shutdown is in progress. Your transaction will be automatically processed when the *PostalOne!* database is back online.

## File Error Messages

System Error (ORA-01034): The *PostalOne!* database is temporarily unavailable. Your transaction will be automatically processed when the *PostalOne!* database is back online.

System Error (ORA-01089): The *PostalOne!* database is being temporarily shut down. Your transaction will be automatically processed when the *PostalOne!* database is back online.

System Error (ORA-01401): A system error was encountered (inserted value too large for column), while processing file md3059.csm (line 1). Please call the *PostalOne!* Customer Care Center and report this error.

System Error (ORA-01480): A system error was encountered (trailing null missing from STRING bind value), while processing file md1796.csm (line 1). Please call the *PostalOne!* Customer Care Center and report this error.

System Error (ORA-01502): The *PostalOne!* database is temporarily unavailable (index or partition of such index is in unstable state). Your transaction will be automatically processed when the *PostalOne!* database is back online.

System Error (ORA-01536): A system error was encountered (space quota exceeded for tablespace 'M\_DYNAMIC1\_D'), while processing file ax02rbal.clr (line 78). Please call the *PostalOne!* Customer Care Center and report this error.

System Error (ORA-01536): The *PostalOne!* database is temporarily unavailable. (The space quota for the segment owner in the tablespace has been exhausted and the operation attempted the creation of a new segment extent in the tablespace) Your transaction will be automatically processed when the *PostalOne!* database is back online.

System Error (ORA-01653): The *PostalOne!* database is temporarily unavailable (failed to allocate extent for the table segment in tablespace). Your transaction will be automatically processed when the *PostalOne!* database is back online.

System Error (ORA-01654): The *PostalOne!* database is temporarily unavailable (failed to allocate extent for the index segment in tablespace). Your transaction will be automatically processed when the *PostalOne!* database is back online.

Data Error (ORA-02291): A Mail.dat relationship error was encountered (no parent record found) while processing file kfs1w205.cqt (line 970). Please correct the Mail.dat file and resubmit the job.

System Error (ORA-03233): The *PostalOne!* database is temporarily unavailable (failed to allocate an extent for the table subpartition segment in tablespace). Your transaction will be automatically processed when the *PostalOne!* database is back online.

System Error (ORA-03234): The *PostalOne!* database is temporarily unavailable (failed to allocate an extent for the index subpartition segment in tablespace). Your transaction will be automatically processed when the *PostalOne!* database is back online.

Error (ORA-XXXXX): An error was encountered (*error text*) while processing this transaction. Please call the *PostalOne!* Customer Care Center and report this error.

Unable to establish a connection with the *PostalOne!* Server.

System Error: An error occurred while receiving the HDR file (\*.hdr) for this transaction. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

System Error: An error occurred while receiving the SEG file (\*.seg) for this transaction. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

System Error: An error occurred while receiving the MPU file (\*.mpu) for this transaction. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

System Error: An error occurred while receiving the MCR file (\*.mcr) for this transaction. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

System Error: An error occurred while receiving the CPT file (\*.cpt) for this transaction. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

## File Error Messages

System Error: An error occurred while receiving the CSM file (\*.csm) for this transaction. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

System Error: An error occurred while receiving the CLR file (\*.clr) for this transaction. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

System Error: An error occurred while receiving the ICL file (\*.icl) for this transaction. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

System Error: An error occurred while receiving the CQT file (\*.cqt) for this transaction. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

System Error: An error occurred while receiving the PQT file (\*.pqt) for this transaction. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

System Error: An error occurred while receiving the WSR file (\*.wsr) for this transaction. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

System Error: An error occurred while receiving the SNR file (\*.snr) for this transaction. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

System Error: An error occurred while receiving the PLR file (\*.plr) for this transaction. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

System Error: An error occurred while receiving the ICR file (\*.icr) for this transaction. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

System Error: An error occurred while receiving the PAR file (\*.par) for this transaction. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

System Error: An error occurred while receiving the MSR file (\*.msr) for this transaction. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

System Error: An error occurred while receiving the MIR file (\*.mir) for this transaction. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

System Error: An error occurred while receiving the SPR file (\*.spr) for this transaction. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

System Error: An error occurred while receiving the SFR file (\*.sfr) for this transaction. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

System Error: An error occurred while reassembling the Mail.dat job on the *PostalOne!* server. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

System Error: Unable to find one of the Mail.dat files for the submitted job. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

System Error: An error occurred while reading one of the files in this Mail.dat job. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

System Error: The *PostalOne!* server is temporarily unavailable. Please retry this transaction in 10 minutes. If the problem persists, please call the *PostalOne!* Customer Care Center.

Error: The *PostalOne!* server failed to receive the complete Mail.dat files for this transaction. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

User Error: The file name used by this Mail.dat job is not allowed by the *PostalOne!* application. Please correct the file name and resubmit the job. For more information, please check the online documentation, or call the *PostalOne!* Customer Care Center.

User Error: The header record for this delete transaction contains non-zero counts for one or more Mail.dat files (except the segment file) in this transaction. Please correct the Mail.dat data and resubmit the job.

User Error: File failed - validation on the client. Please refer to the validation.log file for details. Please resubmit the job after correcting the validation errors.

## File Error Messages

User Error: An attempt was made to connect to the <i>PostalOne!</i> server using an outdated batch process application. Please download the current version of the batch processor application from the <i>PostalOne!</i> website, and then retry this transaction.
Error: 'Customer-ID' information is missing or invalid due to a transmission error. Please resubmit the job. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.
Error: 'Job-ID' information is missing or invalid due to a transmission error. Please resubmit the job. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.
Error: 'Service Name' information is missing or invalid due to a transmission error. Please resubmit the job. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.
Error: 'Service Date' information is missing or invalid due to a transmission error. Please resubmit the job. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.
Error: 'File-ID' information is missing or invalid due to a transmission error. Please resubmit the job. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.
Error: 'Location-seq' information is missing or invalid due to a transmission error. Please resubmit the job. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.
Error: 'Request-type' information is missing or invalid due to a transmission error. Please resubmit the job. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.
Error: 'File-type' information is missing or invalid due to a transmission error. Please resubmit the job. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.
Error: 'Client-type' information is missing or invalid due to a transmission error. Please resubmit the job. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.
Error: 'Username' information is missing or invalid due to a transmission error. Please resubmit the job. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.
Error: 'Password' information is missing or invalid due to a transmission error. Please resubmit the job. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.
Error: 'Filename' information is missing or invalid due to a transmission error. Please resubmit the job. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.
Error: 'Mailing-facility' information is missing or invalid due to a transmission error. Please resubmit the job. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.
Error: 'Provider-code' information is missing or invalid due to a transmission error. Please resubmit the job. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.
Error: 'Checksum' information is missing or invalid due to a transmission error. Please resubmit the job. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.
Error: 'Job-type' information is missing or invalid due to a transmission error. Please resubmit the job. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.
System Error: A system error occurred in the <i>PostalOne!</i> application while requesting a system resource. Please resubmit the job. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.
Error: An unknown file type was sent to the <i>PostalOne!</i> server. Please check your data files and retry this transaction. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.
System Error: A system error occurred in the <i>PostalOne!</i> application while locating a system resource. Please resubmit the job. If the problem persists, please call the <i>PostalOne!</i> Customer Care Center.
User Error: This Mail.dat transaction (Job ID xxxxxxx) does not contain the required xxx file. Please correct the Mail.dat data and resubmit the job.
User Error: The xxx record count indicated in the header record for Job ID xxxxxxx does not match the actual record count in the xxx file. Please correct the Mail.dat data and resubmit the job.

## File Error Messages

User Error: The optional xxx file indicated in the header record for Job ID xxxxxxx, was not found in this Mail.dat job. Please correct the Mail.dat data and resubmit the job.

Unable to establish a connection with the *PostalOne!* Server.

A newer version of the Batch Processor application has been successfully downloaded.

Unable to download a newer version of the Batch Processor application. Please call the *PostalOne!* Customer Care Center and report this error.

Transferring jobs to the *PostalOne!* server...

The Batch Processor application was unable to create the server log file, and will be aborted. Please make sure that sufficient disk space is available, and the application has the appropriate privileges.

The Batch Processor 'remote' test was unsuccessful.

The Batch Processor 'local' test was unsuccessful.

An internal error occurred in the Batch Processor application. Please call the *PostalOne!* Customer Care Center and report this incident.

The Batch Processor application was unable to process the configuration parameters.

A newer version of the Batch Processor application is available on the *PostalOne!* server and will be downloaded.

The Batch Processor application is starting up...

The (encrypted) remote connection test was successful.

The (encrypted) remote connection test was unsuccessful.

The (unencrypted) remote connection test was successful.

The (unencrypted) remote connection test was unsuccessful.

The *PostalOne!* Server is temporarily busy; your request will be automatically resent in five minutes.

Error: The *PostalOne!* Batch Processor application was unable to overwrite the 'daemonclient.jar' file. Please make sure that the file is writeable, or please rename the existing file.

User Error: An attempt was made to connect to the *PostalOne!* server using an outdated batch processor application. Please download the current version of the batch processor application from the *PostalOne!* website, and then retry this transaction.

An error was encountered while processing file: <filename>

The 'RepositoryLocation' parameter was not found in the postal1.ini file. Please verify and correct the contents of the postal1.ini file, and then restart the application.

The 'SuccessfulRepositoryLocation' parameter was not found in the postal1.ini file. Please verify and correct the contents of the postal1.ini file, and then restart the application.

The 'UnsuccessfulRepositoryLocation' parameter was not found in the postal1.ini file. Please verify and correct the contents of the postal1.ini file, and then restart the application.

The 'Delimiter' parameter was not found in the postal1.ini file. Please verify and correct the contents of the postal1.ini file, and then restart the application.

The 'SystemUsername' parameter was not found in the postal1.ini file. Please verify and correct the contents of the postal1.ini file, and then restart the application.

The 'SystemPassword' parameter was not found in the postal1.ini file. Please verify and correct the contents of the postal1.ini file, and then restart the application.

The 'FileType' parameter was not found in the postal1.ini file. Please verify and correct the contents of the postal1.ini file, and then restart the application.

*Obsolete!*

## File Error Messages

### *PostalOne!* Batch Processor

The Batch Processor application terminated under normal conditions.

No updated receipt status available at this time.

Error: Unable to transfer an error message to the *PostalOne!* server while processing file: '<filename>'.  
Waiting for server response...

Received Successful status code from Server

- or -

Received Error Code from *PostalOne!* Server (code = <response code> )

Received Successful status code from Server

Received Error Code from *PostalOne!* Server (responseCode = <response code> )

Error occurred while processing fileset: <filename>

System Error: Error on server while reading in HTTP input. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

System Error: Error on server while creating directories for file storage. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

System Error: None or partial received from input stream. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

System Error: Error on server writing file from input stream to file. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

System Error: Error occurred while assembling split file on server. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

System Error: Error occurred while reassembling the Mail.dat job on the *PostalOne!* server. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

System Error: Server could not validate data from this application. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

User Error: All file-level status flags in Header file are not identical. Please correct the data of the Mail.dat job and resubmit the job.

User Error: At least one file-level status flag in Header file was not accepted. Please correct the data of the Mail.dat job and resubmit the job.

User Error: Record count for a required file is set to '0' in the Header file. Please correct the data of the Mail.dat job and resubmit the job.

User Error: The header record for this delete transaction contains non-zero counts for one or more Mail.dat files (except the segment file) in this transaction. Please correct the data of the Mail.dat job and resubmit the job.

User Error: No update files found for the update job. Please correct the data of the Mail.dat job and resubmit the job.

System Error: An error occurred while reading one of the files in this Mail.dat job. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

### **Applet:**

System Error: The *PostalOne!* server is temporarily unavailable. Please retry this transaction in 10 minutes. If the problem persists, please call the *PostalOne!* Customer Care Center.

### **Batch Processor:**

System Error: The *PostalOne!* server is temporarily unavailable. This transaction will be automatically resent. If the problem persists, please call the *PostalOne!* Customer Care Center.

## File Error Messages

Error: The *PostalOne!* server failed to receive the complete Mail.dat files in this transaction. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

System Error: Client connection timed out. Please retry this transaction. If the problem persists, please call the *PostalOne!* Customer Care Center.

User Error: The file name used by this Mail.dat job is not allowed by the *PostalOne!* application. Please correct the file name and resubmit the job. For more information, please check the online documentation, or call the *PostalOne!* Customer Care Center.

System Error occurred while splitting multiple job files. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

System Error occurred while reading header file (\*.hdr). Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

System Error occurred while compressing file. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

System Error occurred while splitting files. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

Error: The 'Customer-ID' information is missing or invalid due to a transmission error. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

Error: The 'Job-ID' information is missing or invalid due to a transmission error. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

Error: The 'Service Name' information is missing or invalid due to a transmission error. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

Error: The 'Service Date' information is missing or invalid due to a transmission error. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

Error: The 'File-ID' information is missing or invalid due to a transmission error. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

System Error: A communication error occurred (unable to determine the content type values) while processing this transaction. Please resubmit the job. If the problem persists, please call the *PostalOne!* Customer Care Center.

System Error: Server is currently busy.

Internal System Error: <error code>. Please call the *PostalOne!* Customer Care Center and report this error.

**NOTE:** For information on resolving this error, see Section 3.3.2, Typical File Formatting Problems.

The header file contains columns that do not have a status of 'Update'. Please correct the data of the Mail.dat job and resubmit the job.

# Index

**A**  
ASCII .....3, 20, 24, 26

**B**  
batch processor .....2, 10  
    NT .....22  
    prerequisites .....20  
    scheduling .....30  
    testing setup .....30  
    UNIX .....24  
Business Mail Entry (USPS) .....2  
business mailer .....2

**C**  
canceling jobs ..... See deleting jobs  
certificate signing request .....28  
change job transaction ..... 10  
Component ID file .....31  
configuring Mail.dat files, overview .....5  
Container Label file .....32  
Container Quantity file .....32  
Container Status values .....9  
Container Summary file .....31  
Customer Care Center ..... 1, 30

**D**  
deleting jobs .....9  
delimiter  
    in receipt file .....33  
    parameter .....35  
digital certificate  
    adding to keystore .....28  
    obtaining .....28  
    testing .....29

**E**  
electronic collaboration ..... 1  
encryption .....29  
errors  
    common file formatting .....5  
    transfer .....30, 56  
    validation .....18, 52

**F**  
file transfer ..... See *also*, Mail.dat file  
    applet ..... See manual file transfer  
    error messages .....56  
    overview ..... 11

file validator ..... 11  
firewall .....4, 6, 21

**H**  
Header file .....31  
help desk ..... See Customer Care Center

**I**  
IDEAlliance .....2  
    contacting .....4  
Ink Jet/Container Relationship file .....32  
International Container Label file .....32  
Internet connection .....2

**J**  
Java .....21  
Java upload server ..... 10, 33  
job  
    change transaction ..... 10  
    deleting .....9  
    updating .....7  
Job Status page .....20  
JRE .....21  
JVM .....4

**K**  
keystore .....27

**M**  
Mail.dat file  
    and definitions .....31  
    configuring, overview .....5  
    formatting errors .....5  
    processing overview .....10  
    specification .....2, 4  
    transferring  
        in batch .....20  
        manually .....19  
    update, sending .....7  
    upload status .....20  
    validating .....11  
Mailpiece Unit file .....31  
Manifest Individual file .....32  
Manifest Summary file .....32  
manual file transfer .....19  
MPU/Component Relationship file .....31

**O**  
original mailing, sending .....7

<b>P</b>	
Package Label file .....	32
Package Quantity file .....	32
parameters .....	See postal1.ini file
port numbers .....	4, 21
Postage Adjustment file .....	32
postage statement receipt file .....	37
postal1.ini file .....	33, 37
parameters, NT .....	23
parameters, UNIX .....	25
<i>PostalOne!</i> system	
applying to .....	1
overview .....	1
prerequisites .....	3
problem resolution .....	5, 30
processing files .....	10
Provider Code .....	See ULC

<b>R</b>	
receipt file	
content .....	33, 37
layout .....	33, 37
record types .....	33, 42
transfer status .....	35
Record Level Status values .....	8
relationship constraint .....	15
replacing jobs .....	10

<b>S</b>	
scheduling software .....	21
Seed Name file .....	32
Segment file .....	31
Single-piece file .....	32

Special Fees and Charges file .....	32
SSL .....	5, 10, 27

<b>T</b>	
transaction type .....	8
transfer applet .....	See manual file transfer
Transfer Summary .....	19
transferred job, checking .....	20
transferring methods, introduction .....	2
troubleshooting .....	5

<b>U</b>	
ULC .....	5
UNIX .....	21
update	
commands .....	9
rules for .....	8
sending .....	7
upload status .....	20

<b>V</b>	
validation	
overview .....	11
rules .....	12
validation log file .....	18
error codes listing .....	52

<b>W</b>	
Walk Sequence file .....	32

<b>X</b>	
XML .....	3, 20, 24, 26, 33, 43