Pitney Bowes Software

Getting Started With Finalist Guide

Software Release 8.2.0

November 2012

For Linux, Unix, Windows, and z/OS

www.g1.com/support

Report an Error

Pitney Bowes Software
Documentation Team
pbbidocs@pb.com

Table of Contents

GLOSSARY

INDEX

CHAPTER 1

Introduction to Finalist

This chapter provides an introduction to Finalist.

What Is Finalist?

Finalist® is a USPS CASS-certified software application that helps you manage the address information in your customer file(s). Finalist corrects and standardizes address elements in your file including street names, directionals, suffixes, city names, states, ZIP Codes and ZIP + 4 Codes, and then adds the carrier routes and delivery point barcodes you need to maximize your postal discounts.

Finalist Benefits

Some of the benefits available to you when processing with Finalist include:

- Eliminates the expense of developing and maintaining an in-house software program and comprehensive data file
- Reduces your non-deliverable as addressed mail
- Reduces fraud by ensuring accurate and deliverable addresses
- Minimizes "lost" mail - and customers - resulting from incorrect or improperly formatted addresses
- Provides an option to update your address files with corrected information from Finalist processing
- Reduces postage costs by adding ZIP + 4 codes and associated delivery point barcodes to address records to qualify for postage discounts
- Ensures address list accuracy
- Improved demographic information. Adding ZIP + 4 Codes allows you to identify a particular block, post office box, building, apartment, or business location.

How Does Finalist Process Addresses in My File?

Finalist performs the following process on the addresses in your input file:

1. Finalist corrects misspellings in firm, street, and city names.

2. Finalist standardizes address elements such as directionals (NE, West, etc.) and suffixes (Ave, Lane, etc.) in compliance with USPS Coding Accuracy Support System (CASS) regulations.

3. Finalist compares each address to the Finalist data files to verify the accuracy of each address.

4. Finalist corrects errors in the ZIP Code, ZIP + 4 Code, and carrier route codes. These codes are the key to USPS deliverability and minimizing your undeliverable-as-addressed mail. Pitney Bowes Software supplies customers with updated data files on a regular basis.

5. For addresses that are not exact matches to the ZIP + 4 File and would require changes to match the ZIP + 4 File, Finalist can compare the address against the USPS EWS File.

Input

Finalist processing input consists of the following items:

- Your Name-and-Address Input File
- Finalist Data File
- Finalist City/State File
- Additional databases (for example, Delivery Point Validation (DPV), LACS^{Link}, etc.)
- Job control and formatting information

Output

Finalist processing generates the following output:

- **Corrected and standardized addresses** — You determine whether the processing results are written to your input file or to additional output files.
- **Return codes** — You can use the Finalist return codes to analyze addresses that Finalist could not match to USPS data to determine any corrective action.
- **Reports** — These Finalist reports provide information for the processing performed on your input file during the processing run:
  - **Address Detail Report** — The Address Detail Report consists of four sections providing processing information for each input address.
  - **Finalist Batch Report** — The Finalist Batch Report displays all configurable options and statistics generated at the time of the processing run.
  - **USPS Form 3553 (CASS Summary Report)** — With each mailing submitted at an automation-based rate, a mailer must submit a completed USPS Form 3553 (CASS Summary Report). You can submit the original USPS Form or the facsimile generated by your program. Finalist will produce a facsimile of USPS Form 3553 (CASS Summary Report) and partially complete it for you.
- **Error messages** — Finalist error messages assist in troubleshooting any errors you may encounter during processing.

For more detailed information on Finalist output, refer to your *Working With Finalist Guide*.

Finalist Options

Finalist includes options for performing additional processing on your input file.

Table 1: Finalist Options (Part 1 of 2)

Option	Description
Delivery Point Validation (DPV)	USPS CASS regulations require Delivery Point Validation (DPV) processing to generate the USPS Form 3553 (USPS CASS Summary Report). The Finalist Delivery Point Validation (DPV) Option uses DPV data available from the USPS to determine whether an address actually exists. The Delivery Point Validation (DPV) Option can verify the existence of an address to as fine a level as an apartment or suite. You can use the Delivery Point Validation (DPV) Option to ensure the addresses in your address file are actual physical addresses to which the USPS delivers mail.
LACS^{Link}	USPS CASS regulations require LACS^{Link} processing for CASS certification. The USPS LACS^{Link} database contains data on address conversions (for example, an address converted from a rural route/PO box address format to a number/street address format in order to receive 911 emergency response services). The USPS LACS^{Link} database provides mailers with a method for converting these old addresses. The USPS Change of Address (COA) database does not contain these addresses since these changes are actually conversions and not moves resulting in a change of address. Finalist allows you to query the LACS^{Link} database to convert addresses and retrieve LACS^{Link} product name, version, and database date.
Line of Travel (eLOT)	You can use the Line of Travel (eLOT) Option to add eLOT codes to mailings. Line of travel sequence is an option for mailers who prepare carrier route mailings other than high-density/125-piece or saturation mailings. eLOT sequencing is required for Basic Enhanced Carrier Route Standard Mail except automation-compatible, letter-size pieces.
Early Warning System (EWS)	USPS CASS regulations require all CASS-certified software to be able to read the USPS Early Warning System (EWS) File. You can use the Early Warning System (EWS) Option to verify input addresses not found in the current ZIP + 4 File against the USPS EWS File. The USPS postal database may not contain the most current address information. New address information that is in use, but not yet available on the ZIP + 4 File, can now be found as part of the CASS Department's Early Warning System (EWS).

Table 1: Finalist Options (Part 2 of 2)

Option	Description
SuiteLink	USPS CASS regulations require SuiteLink processing to generate the USPS Form 3553 (USPS CASS Summary Report). The USPS SuiteLink database contains data on business addresses that were identified as high-rise default records during CASS processing. Finalist uses the USPS SuiteLink database to correct the secondary (suite) information in business addresses identified in the input file as high-rise default records. Records that have been processed through CASS Certified™ ZIP + 4® matching software and identified as high-rise defaults are potential candidates for SuiteLink processing.
Exceptions Table	You can use the Exceptions Table Option to alter address input fields to enhance the coding ability of Finalist.
ADDRSCAN	You can use the ADDRSCAN function to: • Identify individual components within a multiple-line address record • Format and standardize the address information using United States Postal Service® (USPS®) logic • Return the processed address lines in the order you specify • Identify firm, street, and city lines, so you can utilize the best combination of data from unstructured lists or multi-use lists to reformat and restructure your address files • Pre-process addresses that contain multiple lines or "floating" lines

Finalist Databases

Table 2: Finalist Databases (Part 1 of 2)

Database	Required/ Recommended	Description
CITYFILE	Required	Provides basic address matching.
DATAFILE	Required	Provides ZIP + 4 address matching.
EWSFILE	Recommended	The Early Warning System (EWS) database provides early alerts to address changes that could impact your address file. For example, your input file contains the address 123 MAIN ST. The ZIP4 DATABASE only contains MAIN RD. Since MAIN RD is the only entry in the ZIP + 4 database, without EWS processing, the input address would be changed to MAIN RD. Finalist can read the EWS database to determine that a new address MAIN ST has been created. Finalist does not change the input address to MAIN RD ensuring that mail is not delivered to the wrong location. Each monthly database ships with a copy of the EWSFILE database. However, you are encouraged to obtain the most current information available from the USPS web site. Visit http://ribbs.usps.gov/cassmass/documents/tech_guides/ and look for file EWS002C0.ZIP.
LOTFILE	Recommended	The Enhanced Line Of Travel or eLOT database provides routing information for your coded addresses. While eLOT processing is not required for CASS certification, it is required to obtain discounts. To accomplish this, perform eLOT processing as part of your address hygiene processing or separately as part of your presort processing.
DPVxDB	Required	The Delivery Point Validation (DPV) database provides point specific information about addresses. The ZIP4 databases match addresses to a range. For example, for the address "100-200 N MAIN ST.", DPV further qualifies the address to identify 101 as a valid delivery point where 103 is not. DPV has three (3) formats of its data: • **Full (often called hash)** — Use DPV Full when storage of the databases is the most critical factor. The DPV Full database requires about 620M of disk storage. • **Split** — Use DPV Split for a medium storage factor. The DPV Split file requires about 1.2GB of disk storage. • **Flat** — Use DPV Flat for the largest storage factor. The DPV Flat file requires about 2.1GB of disk storage. **NOTE:** USPS CASS regulations require DPV processing for CASS certification. If you do not perform DPV processing, Finalist does not generate a USPS Form 3553 (CASS Summary Report).
LLKDB	Required	The LACS^{Link} database provides address conversion. For example, the old style address "RR 1 BOX 123" should be converted to "604 S 450 W" for a more accurate delivery of the mailpiece. This is often referred to as the E911 database since it allows emergency personnel to more accurately identify the address location. **NOTE:** USPS CASS regulations require LACS^{Link} processing for CASS certification. If you do not perform LACS^{Link} processing, Finalist does not generate a USPS Form 3553 (CASS Summary Report).

Table 2: Finalist Databases (Part 2 of 2)

Database	Required/ Recommended	Description
SLKDB	Required	The Suite[Link] database provides more accurate matching for firms and businesses. For example, PITNEY BOWES; 2200 WESTERN CT; LISLE IL 60532 is missing the unit (suite) information to accurately deliver the mail. Suite[Link] provides the ability to look into the address file and determine that firm PITNEY BOWES really belongs at a secondary range of STE 100. **NOTE:** USPS CASS regulations require Suite[Link] processing for CASS certification. If you do not perform Suite[Link] processing, Finalist does not generate a USPS Form 3553 (CASS Summary Report).
RDIDB	Recommended	The Residential Delivery Indicator (RDI) Option is designed to identify if an address is a residential (RDI=Y) or a business (RDI not equal to Y) address. The RDI data files are obtained directly from the USPS.

Finalist Keys

Finalist uses a software-based key to license usage. Use one of these processes to enter your Finalist software key:

- Place the Finalist software key in your source file and enter the Finalist software key in PBFN-GCFG-SOFTWAREKEY (cSoftwareKey).
- Enter the Finalist software key in the pbfn.cfg file.
- Use the KeyStore program to store the Finalist software key in a method available to the Finalist engine. This process makes the Finalist software key automatically available to all programs accessing Finalist.

NOTE: The KeyStore program applies to the Finalist software key only. This process does not apply for the LACS^{Link} or DPV keys.

Windows

Keystore.exe is an optional program that allows you to avoid storing your software key in your individual driver code or in individual pbfn.cfg files. KeyStore.exe will generate a keyfile.txt file that can be placed in the folder where you are running Finalist (not the Finalist /bin folder).

KeyStore.exe is a command line program (not a GUI). The syntax of the program is:

```
C:\Program Files\Pitney Bowes\FinalistXXX\bin\keystore.exe <your software key>
```

KeyStore.exe generates a keyfile.txt file that is to be placed in the folder from which you will run Finalist.

If you run KeyStore.exe without a parameter, KeyStore displays the System ID. It is this System ID that is required to generate your Finalist software key.

UNIX

Keystore is an optional program that allows you to avoid storing your software key in your individual driver code or in individual pbfn.cfg files. KeyStore will generate a keyfile.txt file that can be placed in the folder where you are running Finalist (not the Finalist /bin folder).

KeyStore is a command line program (not a GUI). The syntax of the program is:

```
./keystore <your software key>
```

KeyStore generates a keyfile.txt file that is to be placed in the folder from which you will run Finalist.

If you run KeyStore without a parameter, KeyStore displays the System ID. It is this System ID that is required to generate your Finalist software key.

Linux

Keystore is an optional program that allows you to avoid storing your software key in your individual driver code or in individual pbfn.cfg files. KeyStore will generate a keyfile.txt file that can be placed in the folder where you are running Finalist (not the Finalist /bin folder).

KeyStore is a command line program (not a GUI). The syntax of the program is:

```
./keystore <your software key>
```

KeyStore generates a keyfile.txt file that is to be placed in the folder from which you will run Finalist.

If you run KeyStore without a parameter, KeyStore displays the System ID. It is this System ID that is required to generate your Finalist software key.

Mainframe

PGM=KEYSTORE is an optional program (see the Compatibility Interface note below) that allows you to avoid storing your software key in your individual driver code or in individual PBFNCFG files. PGM=KEYSTORE generates a KEYFILE load module that is linked into your Finalist load library.

If you run PGM=KEYSTORE without a SYSIN file, KeyStore displays the System ID. It is this System ID that is required to generate your Finalist software key.

Reference the KEYSTORE member in the FNSOURCE library for sample JCL to run the KEYSTORE program.

Finalist Compatibility Interface

Finalist Compatibility Interface users are **required** to use PGM=KEYSTORE. CICS and IMS transactions use the Compatibility Interface internally and therefore require the Finalist software key to be stored.

CHAPTER 2

CASS Certification Overview

This chapter provides information on the USPS CASS certification process.

What is CASS Certification?

The USPS designed the CASS certification process to test postal coding software and to influence companies to produce high-quality, addressed mail. High-quality addressed mail is mail that contains all appropriate address elements standardized in a manner regulated by the USPS. The CASS certification process tests the accuracy of the five-digit ZIP Codes, ZIP + 4 codes, and Delivery Point Barcodes that appear on mail pieces. The USPS offers postal discounts to those mailers complying with USPS addressing regulations. To qualify for these discounts, a mailer must use a CASS-certified postal coding software package to assign ZIP Codes, ZIP + 4 codes, and Delivery Point Barcodes to mailings. Software vendors supplying postal coding software must pass a test designed by the USPS in order to have their software designated as CASS-certified.

The CASS certification test is divided into two stages. The Stage 1 File is a test with answers. This stage allows processing results to be compared with those provided by the USPS and to resolve any problems before proceeding to Stage 2 testing.

Stage 2 is the actual certification process (graded by the National Customer Support Center). The Stage 2 File is a test without answers. Vendors must process the USPS-supplied file containing 150,000 address records through their software and return the file to the USPS for grading. Vendors must achieve a USPS-specified level of accuracy to be CASS-certified. The figure below details the levels of CASS certification and the accuracy requirement for each level.

Table 1: USPS CASS Certification Levels and Percentages

Certification Level	Required Accuracy Level
ZIP	98.5%
Carrier Route	98.5%
ZIP + 4	98.5%
Delivery Point Barcode	100%
eLOT	100%
Perfect Addresses	100%

CASS certification is valid through the end of the current CASS cycle. Vendors must apply and achieve a specified level of accuracy for each CASS cycle. In addition to the above overall percentages, address matching software must attain a 98.5% rating in all required categories.

Why is CASS Certification Important for Finalist?

The Finalist software is CASS-certified. This feature assures Finalist customers that their mailings meet USPS regulations and qualify for postal discounts achieved through high-quality addressed mail.

Types of Certification

The USPS performs the following types of mailing certifications:

Table 2: USPS Certification Types

Certification Type	Description
CASS Certification	The USPS uses CASS certification to certify postal coding software packages. This type of certification applies to the Finalist software.
Vendor Certification	The software vendor certifies their postal coding software using the Stage 1 and Stage 2 tests. Pitney Bowes Software CASS-certifies every release of Finalist before any release of the software to customers.
Client Certification	A customer may certify their Finalist integration if they wish. If you decide to certify your Finalist integration, Pitney Bowes Software is not responsible for support of your USPS certification process and audits. **NOTE:** Stage test certifications require the use of special static databases. Please contact Technical Support to obtain these databases if you are trying to certify using Finalist.
Mailing List Certification	A mailer can certify a mailing list by processing address files through CASS-certified software (such as Finalist) every six months. This procedure is required for all mailings receiving postal discounts. The USPS requires the USPS Form 3553 (CASS Summary Report) as proof that a mailing is CASS-certified. For more information on the USPS Form 3553 (CASS Summary Report), refer to your *Working With Finalist Guide*.
PAVE Certification	The USPS uses PAVE certification to certify presort software packages.
Z4Change Certification	The USPS uses Z4Change certification to certify ZIP + 4 change software. This type of certification applies to the Pitney Bowes Z4Select software.

What is the Address Management System II (AMS II)?

AMS II is the USPS system that produces the City/State File and ZIP + 4 File. Pitney Bowes Software uses these files to create the Finalist data files.

National Customer Support Center (NCSC)

The NCSC is the USPS customer support center. This support center is located in Memphis, Tennessee. The NCSC includes the four departments described next.

General Customer Service

The General Customer Service department is responsible for:

- Basic Address Information Systems product information
- AIS product and postal guide orders
- Reporting problems in AMS II files

CASS Certification

The CASS Certification Department is responsible for:

- CASS certification
- Z4Change certification
- Ordering Stage 1 and Stage 2 Files
- Any questions regarding certification

AIS File Quality

The AIS File Quality Department is responsible for quality measures for the following files:

- AMS II City/State
- AMS II ZIP + 4
- Z4Change
- ZIPMOVE
- Dropship
- Enhanced Line of Travel (eLOT)

PAVE Certification

The PAVE Certification Department is responsible for PAVE certification.

CHAPTER 3

Delivery Point Validation (DPV) Option

This chapter describes the Finalist Delivery Point Validation (DPV) Option. You can use the DPV Option to reduce your undeliverable mail.

What is the Delivery Point Validation (DPV) Option?

Finalist matches the addresses in your address file against data provided by the United States Postal Service (USPS). The USPS data file consists of low-to-high ranges for streets in the United States. If your input address falls within the low-to-high range for the street specified in your input address, Finalist standardizes the address and assigns the appropriate ZIP Code, ZIP + 4 Code, carrier route code, and delivery point barcode for that range. However, this process does not ensure that the address actually exists or that the USPS actually delivers mail to the address. It only indicates that your input address falls within a known range for the input address street. According to USPS statistics, up to 10% of this coded mail is still undeliverable as addressed.

For example, your input file contains an address "42 Main Street." During processing, Finalist determines that the address "42 Main Street" falls within the known range "00-99 Main Street." Accordingly, Finalist standardizes the "42 Main Street" address and assigns the ZIP Code, ZIP + 4 Code, carrier route code, and delivery point barcode for the known range "00-99 Main Street" to the "42 Main Street" address. However, the "42 Main Street" address belonged to a building that no longer exists. Any mail addressed to "42 Main Street" will be returned to the sender as undeliverable as addressed.

The Finalist Delivery Point Validation (DPV) Option solves this problem by using DPV data available from the USPS to determine whether an address actually exists. The Delivery Point Validation (DPV) Option can verify the existence of an address to as fine a level as an apartment or suite. You can use the Delivery Point Validation (DPV) Option to ensure the addresses in your address file are actual physical addresses to which the USPS delivers mail.

NOTE: USPS CASS regulations require Delivery Point Validation (DPV) processing for CASS certification and to generate the USPS Form 3553 (USPS CASS Summary Report).

Can I Use the Delivery Point Validation (DPV) Option?

The DPV Option is available to all Finalist customers.

Why Should I Use the DPV Option?

Some of the benefits available to Finalist customers through DPV processing are described below.

- Ensures compliancy with USPS CASS regulations.

- Ensures preparedness for future USPS CASS regulations. In future CASS cycles, the USPS may expand DPV processing requirements.
- Reduces undeliverable mail.
- Decreases cost of postage on returned mail.
- Decreases processing costs for mail that cannot be delivered.
- Improves delivery of targeted marketing mailings.

How Do I Install the DPV Option?

The DPV Option is installed as part of your standard Finalist installation. The distribution media sent to you in your release package contains all data needed to perform DPV processing. After installing Finalist, you must activate the DPV Option to perform DPV processing.

How Do I Activate DPV Processing?

To activate DPV processing, you will need to define the fields listed below.

- **Delivery Point Validation** — This field, in effect, turns on DPV processing.
- **DPV Filepath** — Defines the location of DPV files. If you are processing with the DPV Flat File, this field contains the path to the DPV Flat File.
- **DPVKey** — Ensures you are in compliance with your license agreement.
- **DPV Buffer Size** — Specifies the memory to be used by DPV processing. A small amount of memory is suggested for on-line transactions. A larger amount of memory is suggested for batch processing of a large number of addresses.

Methods for Activating DPV Processing

To activate DPV processing, use one of the following methods.

- **pbfn.cfg Configuration File** — If you prefer to configure your Finalist installation using global settings, you can define the DPV fields in your pbfn.cfg configuration file. For more information on the pbfn.cfg file, refer to your *Working With Finalist Guide*.
- **PBFNSetupDef Structure** — If you prefer to configure your Finalist installation using the Finalist structures, you can define the DPV fields in the PBFNSetupDef structure. For more information on the PBFNSetupDef structure, refer to your *Finalist Developer's Reference Guide*.

- **Workbench or Lookup Tool** — If you prefer to configure your Finalist installation using the Finalist Workbench or Lookup Tool GUI screens, you can define the DPV fields in the **Product Tab** on the *PBFN Config Setting* dialog box. For more information on the Finalist Workbench and Lookup Tool, refer to your *Working With Finalist Guide*.
- **Compatibility Interface (CI)** — If you prefer to configure your Finalist installation using the CI, you can define the DPV fields in the Finalist call area. For more information on the CI, refer to your *Finalist Developer's Reference Guide*.

Each method for activating DPV processing is described in the following sections.

Using pbfn.cfg to Activate DPV Processing

To activate DPV processing in the pbfn.cfg file, complete the following fields in your pbfn.cfg file.

Table 1: pbfn.cfg File — DPV Settings (Part 1 of 3)

pbfn.cfg Field	Description
Delivery Point Validation	Indicate whether to perform Delivery Point Validation (DPV) processing: NOTE: The USPS CASS regulations require Delivery Point Validation (DPV) processing to generate the USPS Form 3553 (USPS CASS Summary Report). • OFF — Do not perform Delivery Point Validation (DPV) processing. • ON — Perform Delivery Point Validation (DPV) processing using the DPV Full database (dpvh.db). • MEM — Perform Delivery Point Validation (DPV) processing with the DPV Full database (dpvh.db) fully in memory for improved performance. NOTE: MEM is converted to ON with a buffersize (cDPVBufSize) of H (huge). • SPL — Perform Delivery Point Validation (DPV) processing using the DPV Split database (dpvs.db). Split File processing separates the DPV Data File into 100 smaller files based on the first two digits of the ZIP Code. The Split File segment associated with the first two digits of the ZIP Code is loaded into memory. If you sort your mailing file by ZIP Code, you can bring the relevant portion of the DPV file into memory. This process reduces the number of I/O requests that normally occurs when you use the full DPV Data File. Use this option if your file is sorted by ZIP Code. This is the optimal solution for Finalist customers having at least 100MB, but no more than 650MB, of RAM/Memory. • FLT — Perform Delivery Point Validation (DPV) processing using the DPV Flat database (dpv.db). Use the DPV Flat File for improved batch performance. This file is in excess of 2.1GB and not dependent on RAM/Memory capacity. Input files should be sorted by ZIP Code. This is an optimal solution if you have less than 100MB of RAM/Memory available. • blank — Defaults to OFF.
DPV Filepath	Specify the Delivery Point Validation (DPV) file path.
DPVKey	Enter the Delivery Point Validation (DPV) activation key.

Table 1: pbfn.cfg File — DPV Settings (Part 2 of 3)

pbfn.cfg Field	Description
Delivery Point Validation Tie Break	The USPS allows DPV processing to be used as a tie breaker for matching inexact street records. If only one of the records in a tie is delivery point validated, a match is allowed to the inexact record. When processing results in an inexact match due to the input address directional, DPV processing can be used as a tie breaker if only one of the records is found to be delivery point validated and the delivery point validated record does not violate the cardinal direction rule. **NOTE:** The USPS CASS regulations require DPV Tie Break processing to generate the USPS Form 3553 (USPS CASS Summary Report). Indicate whether to perform DPV Tie Break processing: • **OFF** — Do not perform DPV Tie Break processing. • **ON** — Perform DPV Tie Break processing. The DPV Tie Break Option can increase your matching percentages but can also negatively impact performance. For memory loading options, refer to the "DPV Buffer Size" field description in this table. • blank — Defaults to **ON**. **NOTE:** The default value "ON" only applies when "Delivery Point Validation" is defined as ON, MEM, SPL, or FLT.
DPV Shutdown Indicator	Indicate the action to take when encountering a DPV False Positive (Seed) violation during the processing run: • **W** — Issue warning message when a DPV False Positive (Seed) violation has been encountered. Processing continues but DPV processing is disabled. • **S** — Stop processing when encountering a DPV False Positive (Seed) violation. Finalist issues an error message, stops processing remaining addresses, and exits the program. • blank — Defaults to **W**. If a DPV seed is encountered and **W** is set, CASS processing is turned off. CASS processing is turned off for any jobs submitted after the DPV False Positive (Seed) violation was encountered. For the job that encountered the DPV False Positive (Seed) violation, the CASS statement will be provided showing the number of records that were DPV confirmed up until the point of the DPV False Positive (Seed) violation. No records will be DPV confirmed after the DPV False Positive (Seed) violation occurred. Any jobs submitted before the DPV False Positive (Seed) violation was encountered will run through to completion, including DPV processing for each file. This occurs even if the jobs do not complete processing until after the time the DPV False Positive (Seed) violation was found in the job that first encountered the DPV False Positive (Seed) violation. Finalist generates the USPS Form 3553 (CASS Summary Report) for these jobs. For any job set to perform CASS processing and submitted after a DPV False Positive (Seed) violation occurs, Finalist generates an initialization error and stops processing. **NOTE:** The Shutdown Indicator applies to batch processing only. If your DPV key is NCOA or "No Stop", this option is ignored.

Table 1: pbfn.cfg File — DPV Settings (Part 3 of 3)

pbfn.cfg Field	Description
DPV No-Stat Table	The USPS has added a No-Stat Table for DPV processing to identify deliveries that are not valid for Computerized Delivery Sequence (CDS) pre-processing. Indicate whether to use the No-Stat Table and return the proper No-Stat code to the output: • **OFF** — Do not perform No-Stat Table processing. • **ON** — Perform No-Stat Table processing. For memory loading options, refer to the "DPV Buffer Size" field description in this table. • blank — Defaults to **OFF**.
DPV Vacant Table	The USPS has added a Vacant Table for DPV processing. This table identifies delivery addresses that have been active in the past but, according to USPS data, have not been occupied within the last 90 days. Indicate whether to use the Vacant Table and return the proper Vacant code to the output: • **OFF** — Do not perform Vacant Table processing. • **ON** — Perform Vacant Table processing. For memory loading options, refer to the DPV Buffer Size field description in this table. • Blank — Defaults to **OFF**.
Commercial Mail Validation	Indicate whether to perform CMRA processing. Private companies offering mailbox rental services to individuals and businesses are Commercial Mail Receiving Agents (CMRA). • **OFF** — Do not perform CMRA processing. • **ON** — Perform CMRA processing. • blank — Defaults to **OFF**.
DPV Buffer Size	Indicates the memory model for DPV processing: • **P** — Pico. Stores no data in memory. No tables or indexes are loaded. • **U** — Ultra-small. Stores no data in memory. Partial indexes are loaded. • **S** — Small • **M** — Medium • **L** — Large • **H** — Huge. Stores all data in memory. • blank — Defaults to **M**. Values from releases prior to the Finalist 8.0.0 release are accepted as: • If you specified 0, M is substituted. • If you specified 1, U is substituted • If you specified a value greater than 1 but less than or equal to 30, S is substituted. • If you specified a value greater than 30 but less than or equal to 500, M is substituted. • If you specified a value greater than 500, but less than or equal to 900, L is substituted. • If you specified a value greater than 900, H is substituted.

Using the PBFNSetupDef Structure to Activate DPV Processing

To activate DPV processing in the PBFNSetupDef structure, complete the following fields in the PBFNSetupDef structure.

Table 2: PBFNSetupDef Structure — DPV Settings (Part 1 of 3)

PBFNSetupDef Field	Description
cAssignDPV	Indicate whether to perform Delivery Point Validation (DPV) processing: **NOTE:** The USPS CASS regulations require Delivery Point Validation (DPV) processing to generate the USPS Form 3553 (USPS CASS Summary Report). • **OFF** — Do not perform Delivery Point Validation (DPV) processing. • **ON** — Perform Delivery Point Validation (DPV) processing using the DPV Full database (dpvh.db). • **MEM** — Perform Delivery Point Validation (DPV) processing with the DPV Full database (dpvh.db) fully in memory for improved performance. This option requires a minimum of 650MB of RAM/Memory. **NOTE:** MEM is converted to ON with a buffersize (cDPVBufSize) of H (huge). • **SPL** — Perform Delivery Point Validation (DPV) processing using the DPV Split database (dpvs.db). Split File processing separates the DPV Data File into 100 smaller files based on the first two digits of the ZIP Code. The Split File segment associated with the first two digits of the ZIP Code is loaded into memory. If you sort your mailing file by ZIP Code, you can bring the relevant portion of the DPV file into memory. This process reduces the number of I/O requests that normally occurs when you use the full DPV Data File. Use this option if your file is sorted by ZIP Code. This is the optimal solution for Finalist customers having at least 100MB, but no more than 650MB, of RAM/Memory. • **FLT** — Perform Delivery Point Validation (DPV) processing using the DPV Flat database (dpv.db). Use the DPV Flat File for improved batch performance. This file is in excess of 2.1GB and not dependent on RAM/Memory capacity. Input files should be sorted by ZIP Code. This is an optimal solution if you have less than 100MB of RAM/Memory available. • blank — Defaults to **OFF**.
cDPVFilePath	Specify the Delivery Point Validation (DPV) file path.
cDPVKeyName	Enter the Delivery Point Validation (DPV) activation key.

Table 2: PBFNSetupDef Structure — DPV Settings (Part 2 of 3)

PBFNSetupDef Field	Description
cAssignDPVTie	The USPS allows DPV processing to be used as a tie breaker for matching inexact street records. If only one of the records in a tie is delivery point validated, a match is allowed to the inexact record. When processing results in an inexact match due to the input address directional, DPV processing can be used as a tie breaker if only one of the records is found to be delivery point validated. **NOTE:** The USPS CASS regulations require DPV Tie Break processing to generate the USPS Form 3553 (USPS CASS Summary Report). Indicate whether to perform DPV Tie Break processing: • **OFF** — Do not perform DPV Tie Break processing. • **ON** — Perform DPV Tie Break processing. The DPV Tie Break Option can increase your matching percentages but can also negatively impact performance. For memory loading options, refer to the "cDPVBufSize" field description in this table. • blank — Defaults to **ON**. **NOTE:** The default value "ON" only applies when cAssignDPV is defined as ON, MEM, SPL, or FLT.
cDPVShutdownIndicator	Indicate the action to take when encountering a DPV False Positive (Seed) violation during the processing run: • **W** — Issue warning message when a DPV False Positive (Seed) violation has been encountered. Processing continues but DPV processing is disabled. • **S** — Stop processing when encountering a DPV False Positive (Seed) violation. Finalist issues an error message, stops processing remaining addresses, and exits the program. • blank — Defaults to **W**. If a DPV seed is encountered and **W** is set, CASS processing is turned off. CASS processing is turned off for any jobs submitted after the DPV False Positive (Seed) violation was encountered. For the job that encountered the DPV False Positive (Seed) violation, the CASS statement will be provided showing the number of records that were DPV confirmed up until the point of the DPV False Positive (Seed) violation. No records will be DPV confirmed after the DPV False Positive (Seed) violation occurred. Any jobs submitted before the DPV False Positive (Seed) violation was encountered will run through to completion, including DPV processing for each file. This occurs even if the jobs do not complete processing until after the time the DPV False Positive (Seed) violation was found in the job that first encountered the DPV False Positive (Seed) violation. Finalist generates the USPS Form 3553 (CASS Summary Report) for these jobs. For any job set to perform CASS processing and submitted after a DPV False Positive (Seed) violation occurs, Finalist generates an initialization error and stops processing. **NOTE:** The Shutdown Indicator applies to batch processing only. If your DPV key is NCOA or "No Stop", this option is ignored.

Table 2: PBFNSetupDef Structure — DPV Settings (Part 3 of 3)

PBFNSetupDef Field	Description
cAssignDPVVacant	The USPS has added a Vacant Table for DPV processing. This table identifies delivery addresses that have been active in the past but, according to USPS data, have not been occupied within the last 90 days. Indicate whether to use the Vacant Table and return the proper Vacant code to the PBFNProcessDataDef structure: • **OFF** — Do not perform Vacant Table processing. • **ON** — Perform Vacant Table processing. For DPV memory loading options, refer to the "cDPVBufSize" field description in this table. • Blank — Defaults to **OFF**.
cAssignDPVNoStat	The USPS has added a No-Stat Table for DPV processing to identify deliveries that are not valid for Computerized Delivery Sequence (CDS) pre-processing. Indicate whether to use the No-Stat Table and return the proper No-Stat code to the output: • **OFF** — Do not perform No-Stat Table processing. • **ON** — Perform No-Stat Table processing. For DPV memory loading options, refer to the "cDPVBufSize" field description in this table. • blank — Defaults to **OFF**.
cAssignCMRA	Indicate whether to perform CMRA processing. Private companies offering mailbox rental services to individuals and businesses are Commercial Mail Receiving Agents (CMRA). • **OFF** — Do not perform CMRA processing. • **ON** — Perform CMRA processing. • blank — Defaults to **OFF**.
cDPVBufSize	Indicates the memory model for DPV processing: • **P** — Pico. Stores no data in memory. No tables or indexes are loaded. • **U** — Ultra-small. Stores no data in memory. Partial indexes are loaded. • **S** — Small • **M** — Medium • **L** — Large • **H** — Huge. Stores all data in memory. • blank — Defaults to **M**. Values from releases prior to the Finalist 8.0.0 release are accepted as: • If you specified 0, M is substituted. • If you specified 1, U is substituted • If you specified a value greater than 1 but less than or equal to 30, S is substituted. • If you specified a value greater than 30 but less than or equal to 500, M is substituted. • If you specified a value greater than 500, but less than or equal to 900, L is substituted. • If you specified a value greater than 900, H is substituted.

Using the Finalist Workbench to Activate DPV Processing

To activate DPV processing in the Finalist Workbench:

1. Launch the Finalist Workbench.
2. From the **Tools** menu, select **PBFN Setup**.
3. Select the **Product** tab on the *PBFN Config Setting* dialog box.

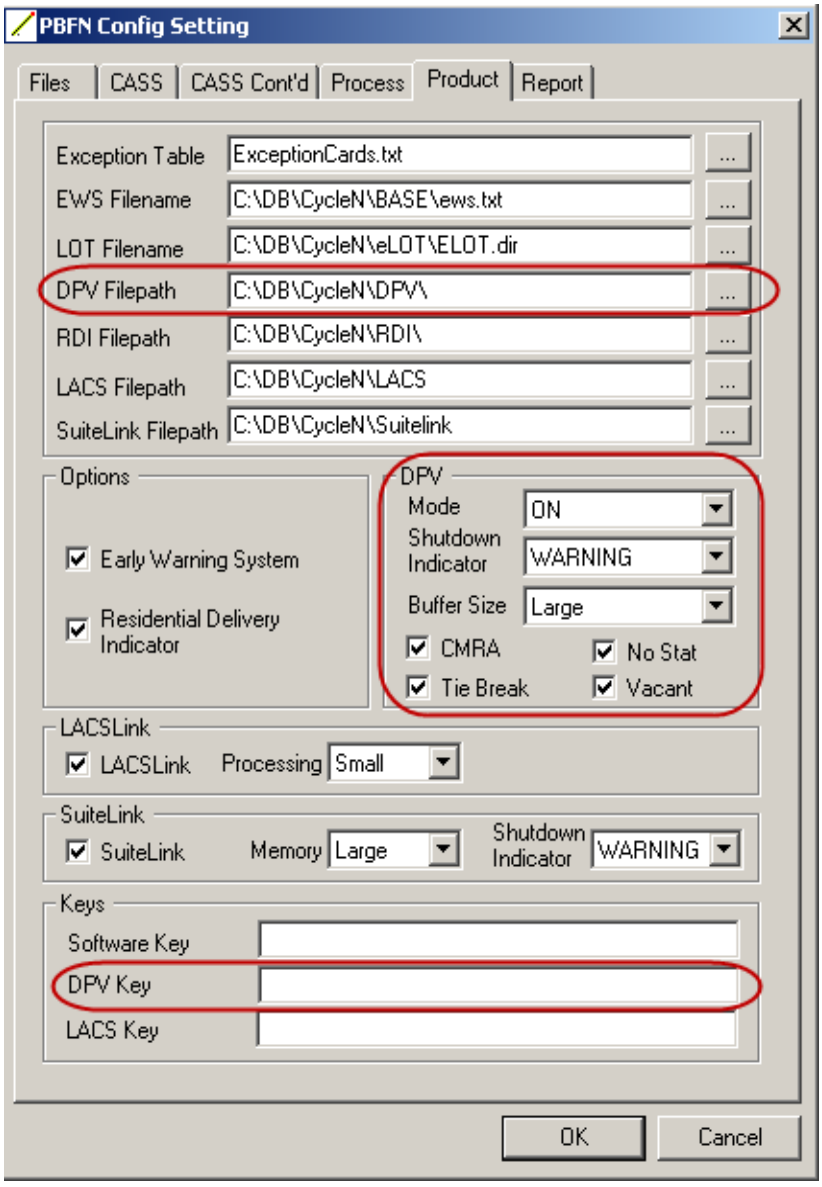

Figure 1: PBFN Config Setting Dialog Box — Product Tab

4. Complete the following fields on the **Product** Tab.

Table 3: PBFN Config Setting Dialog Box — Product Tab (Part 1 of 3)

Field	Description
DPV Filepath	Specify the Delivery Point Validation (DPV) file path or click on the **Browse** button (located to the right of the field) to open a dialog box and point to the folder (path) that contains the databases.
Mode	Indicate whether to perform Delivery Point Validation (DPV) processing: **NOTE:** The USPS CASS regulations require Delivery Point Validation (DPV) processing to generate the USPS Form 3553 (USPS CASS Summary Report). • **OFF** — Do not perform Delivery Point Validation (DPV) processing. • **ON** — Perform Delivery Point Validation (DPV) processing using the DPV Full database (dpvh.db). • **MEM** — Perform Delivery Point Validation (DPV) processing with the DPV Full database (dpvh.db) fully in memory for improved performance. This option requires a minimum of 650MB of RAM/Memory. **NOTE:** MEM is converted to ON with a buffer size of H (huge). • **SPL** — Perform Delivery Point Validation (DPV) processing using the DPV Split database (dpvs.db). Split File processing separates the DPV Data File into 100 smaller files based on the first two digits of the ZIP Code. The Split File segment associated with the first two digits of the ZIP Code is loaded into memory. If you sort your mailing file by ZIP Code, you can bring the relevant portion of the DPV file into memory. This process reduces the number of I/O requests that normally occurs when you use the full DPV Data File. Use this option if your file is sorted by ZIP Code. This is the optimal solution for Finalist customers having at least 100MB, but no more than 650MB, of RAM/Memory. • **FLT** — Perform Delivery Point Validation (DPV) processing using the DPV Flat database (dpv.db). Use the DPV Flat File for improved batch performance. This file is in excess of 2.1GB and not dependent on RAM/Memory capacity. Input files should be sorted by ZIP Code. This is an optimal solution if you have less than 100MB of RAM/Memory available. • blank — Defaults to **OFF**.

Table 3: PBFN Config Setting Dialog Box — Product Tab (Part 2 of 3)

Field	Description
Shutdown Indicator	Indicate the action to take when encountering a DPV False Positive (Seed) violation during the processing run: • **WARNING** — Issue warning message when a DPV False Positive (Seed) violation has been encountered. Processing continues but DPV processing is disabled. • **SHUTDOWN** — Stop processing when encountering a DPV False Positive (Seed) violation. Finalist issues an error message, stops processing succeeding addresses, and exits the program. • blank — Defaults to **WARNING**. If a DPV False Positive (Seed) violation is encountered and **WARNING** is set, CASS processing is turned off. CASS processing is turned off for any jobs submitted after the DPV False Positive (Seed) violation was encountered. For the job that encountered the DPV False Positive (Seed) violation, the CASS statement will be provided showing the number of records that were DPV confirmed up until the point of the DPV False Positive (Seed) violation. No records will be DPV confirmed after the DPV False Positive (Seed) violation occurred. For any job set to perform CASS processing and submitted after a the DPV False Positive (Seed) violation occurs, Finalist generates an initialization error and stops processing. Any jobs submitted before the DPV False Positive (Seed) violation was encountered will run through to completion, including DPV processing for each file. This occurs even if the jobs do not complete processing until after the time the DPV False Positive (Seed) violation was found in the job that first encountered the DPV False Positive (Seed) violation. Finalist generates the USPS Form 3553 (CASS Summary Report) for these jobs. **NOTE:** The Shutdown Indicator applies to batch processing only. If your DPV key is NCOA or "No Stop", this option is ignored.

Table 3: PBFN Config Setting Dialog Box — Product Tab (Part 3 of 3)

Field	Description
Buffer Size	Indicates the memory model for DPV processing: • **Pico** — Pico. Stores no data in memory. No tables or indexes are loaded. • **Ultra** — Ultra-small. Stores no data in memory. Partial indexes are loaded. • **Small** — Small • **Medium** — Medium • **Large** — Large • **Huge** — Huge. Stores all data in memory. • blank — Defaults to **Medium**. Values from releases prior to the Finalist 8.0.0 release are accepted as: • If you specified 0, Medium is substituted. • If you specified 1, Ultra is substituted • If you specified a value greater than 1 but less than or equal to 30, Small is substituted. • If you specified a value greater than 30 but less than or equal to 500, Medium is substituted. • If you specified a value greater than 500, but less than or equal to 900, Large is substituted. • If you specified a value greater than 900, Huge is substituted.
CMRA	Check the **CMRA** box to perform CMRA processing. Private companies offering mailbox rental services to individuals and businesses are Commercial Mail Receiving Agents (CMRA). Defaults to **OFF**.
No Stat	The USPS has added a No-Stat Table for DPV processing to identify deliveries that are not valid for Computerized Delivery Sequence (CDS) pre-processing. Check the **No Stat** box to use the USPS No-Stat Table for DPV processing and return the proper No-Stat code to the output file. Defaults to **OFF**.
Tie Break	The USPS allows DPV processing to be used as a tie breaker for matching inexact street records. If only one of the records in a tie is delivery point validated, a match is allowed to the inexact record. When processing results in an inexact match due to the input address directional, DPV processing can be used as a tie breaker if only one of the records is found to be delivery point validated and the delivery point validated record does not violate the cardinal direction rule. Check the **Tie Break** box to perform DPV Tie Break processing. **NOTE:** The USPS CASS regulations require DPV Tie Break processing to generate the USPS Form 3553 (USPS CASS Summary Report).
Vacant	The USPS has added a Vacant Table for DPV processing. This table identifies delivery addresses that have been active in the past but, according to USPS data, have not been occupied within the last 90 days. Check the **Vacant** box to use the Vacant Table and return the proper Vacant code to the output file. Defaults to **OFF**.
DPV Key	Enter the Delivery Point Validation (DPV) activation key.

5. Click **OK** to save your settings.

Using the Compatibility Interface (CI) to Activate DPV Processing

To activate DPV processing using the CI, define the appropriate field for your platform in the Finalist call area.

Table 4: Compatibility Interface (CI) - DPV Settings (Part 1 of 2)

COBOL Field Name/ C Field Name	Field Description
FINAL-DPV-OPT caDPV	Determines whether Finalist performs Delivery Point Validation (DPV) Option processing. • **N** — No DPV processing. • **Y** — DPV processing using DPV Full File. • **S** — DPV processing using DPV Split File. • **F** — DPV processing using DPV Flat File. • blank — Defaults to **N**.

Table 4: Compatibility Interface (CI) - DPV Settings (Part 2 of 2)

COBOL Field Name/ C Field Name	Field Description
FINAL-DPV-BUFFER-SIZE caDPVbuf	Indicates the memory model for DPV processing: • **P** — Pico. Stores no data in memory. No tables or indexes are loaded. • **U** — Ultra-small. Stores no data in memory. Partial indexes are loaded. • **S** — Small • **M** — Medium • **L** — Large • **H** — Huge. Stores all data in memory. • blank — Defaults to **M**. Values from releases prior to the Finalist 8.0.0 release are accepted as: • If you specified 0, M is substituted. • If you specified 1, U is substituted • If you specified a value greater than 1 but less than or equal to 30, S is substituted. • If you specified a value greater than 30 but less than or equal to 500, M is substituted. • If you specified a value greater than 500, but less than or equal to 900, L is substituted. • If you specified a value greater than 900, H is substituted.
FINAL-DPV-SI caDPVSDI	Determines the action to take when encountering a DPV False Positive (Seed) violation during the processing run. • **W** — Issue warning message when a DPV False Positive (Seed) violation has been encountered. Processing continues but DPV processing is disabled. • **S** — Stop processing when encountering a DPV False Positive (Seed) violation. Finalist issues an error message, stops processing succeeding addresses, and exits the program. blank — Defaults to **W**.

The CI does not support turning on or off these options:

- CMRA processing
- No Stat processing
- Tie Break processing
- DPV Vacant Table processing

Using DPV Processing With User-Written Drivers

To enable DPV processing from a user-written driver using the PBFN APIs, follow the directions below for your programming language.

For COBOL, follow these steps:

1. Use the updated PBFN-GCFG-SETUP structure from PBFNGCFG copybook.
2. Include the following statement before your PBFNInit function call to enable DPV processing:

   ```
   SET PBFN-GCFG-ASSIGNDPV       TO 'XXX'
   ```

   Replace “XXX” with the appropriate value for the type of DPV processing to perform. For more information on DPV processing type, refer to the “cAssignDPV” field description in Table 2, “PBFNSetupDef Structure — DPV Settings,” on page 29.

For C, follow these steps:

1. Use the updated PBFNSetupDefinition structure from pbfnxdef.h.
2. Include the following statement before your PBFNInit function call to enable DPV processing:

   ```
   strcpy (cAssignDPV, "XXX");
   ```

   Replace “XXX” with the appropriate value for the type of DPV processing to perform. For more information on DPV processing type, refer to the “cAssignDPV” field description in Table 2, “PBFNSetupDef Structure — DPV Settings,” on page 29.

DPV Structures

This section provides an overview of the DPV structures. For detailed information on these structures refer to Chapter 2, Structures and Constants in your *Finalist Developer's Reference Guide*. Table 5 describes the DPV structures.

Table 5: DPV Structures

Structure	Description
PBFNDPVDetailDef	This structure returns DPV False Positive (Seed) Table detail record.
PBFNDPVHdrDef	PBFNDPVHdrDef returns the Delivery Point Validation (DPV) Option Header record.
PBFNDPVStatsDef	You can use this structure to find DPV processing statistics.
USPSDetailDef	This structure returns the detail data required by the USPS for each record creating a Delivery Point Validation (DPV) Option False Positive (Seed) Table violation.
USPSDPVHdrDef	Use this structure to return the header data required by the USPS for Delivery Point Validation (DPV) Option False Positive (Seed) Table violations.

Other Structures Containing DPV Information

The structures listed below include data to facilitate DPV processing. For detailed information on these structures, refer to your *Finalist Developer's Reference Guide*.

- PBFNIMSSetupDef
- PBFNParsedAdrAltDef
- PBFNParsedAdrDef (includes DPV indicators and footnotes)
- PBFNProcessDataAltDef
- PBFNProcessDataDef

NOTE: PBFNProcessDataDef includes the DPV indicators and contains the cDPVFootnote and sDPVFootnoteLen fields. These fields define the footnote codes returned during Delivery Point Validation (DPV) Option processing. These codes provide information on your processed input address.

- PBFNSetupDef (includes three DPV setup parameters)

DPV Output

The following sections describe Delivery Point Validation (DPV) Option output.

DPV Error Messages

The following table describes the Delivery Point Validation (DPV) Option error messages.

Table 6: Delivery Point Validation (DPV) Option Error Messages

Error Message	Description
00005 - DPV Processing Error Problem opening XXXXXXXX file	Cannot open DPV file. Verify path or DDNAME.
20100 - DPV Processing Error Call Product tech support Case 0 DPV0	Cannot open the security file.
20101 - DPV Processing Error Call Product tech support Case 0 DPV1	Cannot read the security file.
20111 - DPV Processing Error Call Product tech support Terminate Case 1 DPV1	Cannot open the security file.
20112 - DPV Processing Error Call Product tech support Terminate Case 1 DPV2	Cannot read the security file.
20113 - DPV Processing Error Call Product tech support Terminate Case 1 DPV3	Cannot open the security file for write processing.
20120 - DPV Processing Error File is not same month (yymm) as ZIP+4 file (yymm)	The date of the Delivery Point Validation (DPV) Option data does not match the date of the Zip4us.dir File. Each month of DPV data corresponds to a specific month of ZIP + 4 data (i.e., the September DPV data must be processed with the September ZIP + 4 data).
20121 - DPV Processing Error Call Product tech support Case 2 DPV1	Invalid security file - internal date.
20122 - DPV Processing Error Call Product tech support Case 2 DPV2	Invalid security file - internal check digit.
20123 - DPV Processing Error Call Product tech support Case 2 DPV3	Invalid security file - internal check digit.
20124 - DPV Processing Error Call Product tech support Case 2 DPV4	Seed violation occurred. Send data to the USPS and obtain a new key from the Pitney Bowes Software web site at www.g1.com/support.
20125 - DPV Processing Error Call Product tech support Case 2 DPV5	Cannot open the security file for write processing.

Table 6: Delivery Point Validation (DPV) Option Error Messages

Error Message	Description
20126 - DPV Processing Error Call Product tech support Case 2 DPV6	Invalid security file - sequence number.
20127 - DPV Processing Error Call Product tech support Case 2 DPV7	Invalid security file - unknown error.

DPV Return Information

The DPV processing return codes and footnote codes can be found in the PBFNProcessDataDef structure (or PBFNProcessDataAltDef for languages like COBOL that need to use character arrays in place of pointers for data fields). The following table displays the DPV return flags, DPV No-Stat indicator, and DPV Vacant Table indicator.

Table 7: DPV Return Indicators

Field	Description
cDPVFlags	Character array containing the returned DPV indicators. • Byte 1 (DPV) – **N** — Not a valid delivery point. The USPS cannot deliver mail to this address. – **Y** — Delivery point validated. Primary range and secondary range (when present) are valid. The USPS can deliver mail to this address. – **S** — Valid primary range. Secondary range is present but is not confirmed. The USPS can deliver mail to this address. – **D** — Valid primary range. Secondary range is missing. The USPS can deliver mail to this address. • Byte 2 (CMRA) – **Y** — The address is a valid Commercial Mail Receiving Agent (CMRA). – **N** — The address is a confirmed delivery point but is not a valid CMRA. – Blank — This field is blank if the address is not a confirmed delivery point. • Byte 3 (False Positive Flag) – **Y** — The address is not a confirmed delivery point and a positive response is received from the False Positive File. – **N** — The address is not a confirmed delivery point and a negative response is received from the False Positive File. – Blank — The False/Positive Table was not queried. The address is a confirmed delivery point.
cDPVNoStatFound	Character array indicating DPV No-Stat Table status. • **Y** — Found in the DPV No-Stat Table. • **N** — Not found in the DPV No-Stat Table.
cDPVVacantFound	DPV Vacant Table status indicator. • **Y** — Found in the DPV Vacant Table. • **N** — Not found in the DPV Vacant Table.

The following table displays the DPV footnote codes (cDPVFootnote).

Table 8: DPV Footnote Codes

Field	Description
cDPVFootnote	Field containing the returned Delivery Point Validation (DPV) Option footnote field. This field defines the standard footnote codes returned during Delivery Point Validation (DPV) Option processing. Finalist returns up to six two-byte footnote codes for each address. • **AA** — Input address matched to the ZIP + 4 File. • **A1** — Input address did not match to the ZIP + 4 File. • **BB** — Input address matched to DPV (all components). • **CC** — Input address primary number matched to DPV but secondary number did not match (present but invalid). • **F1** — Input address matched to a military ZIP Code. • **G1** — Input address matched to a General Delivery address. • **N1** — Input address primary number matched to DPV but address is missing secondary number. • **M1** — Input address primary number missing. • **M3** — Input address primary number is invalid. • **P1** — Input address missing PO, RR, or HC Box number. • **P3** — Input address PO, RR, or HC box number invalid. • **RR** — Input address matched to CMRA. • **R1** — Input address matched to CMRA but secondary number is not present. • **U1** — Input address matched to a unique ZIP Code.

DPV Information on Finalist Batch Report

The Finalist Batch Report provides statistics for DPV and CMRA processing. For detailed information on the Finalist Batch Report, refer to your *Working With Finalist Guide*.

DPV Information on USPS Form 3553 (CASS Summary Report)

The C. Output section of the USPS Form 3553 (CASS Summary Report) includes information for Delivery Point Validation (DPV) processing. For detailed information on the USPS Form 3553 (CASS Summary Report), refer to your *Working With Finalist Guide*.

```
== C. OUTPUT ========================================================================
OUTPUT RATING          |VALIDATION PERIOD |OUTPUT RATING     |VALIDATION PERIOD
            TOTAL CODED|FROM    TO        |       TOTAL CODED|FROM    TO
-----------------------|------------------|------------------|------------------
a. ZIP+4/DPV CONFIRMED |XX/XX/XXXX        |d. 5-DIGIT CODED  |XX/XX/XXXX
               1234567 |        XX/XX/XXXX|         1234567  |        XX/XX/XXXX
-----------------------|------------------|------------------|------------------
b. Z4CHANGE PROCESSED  |//////////////////|e. CR RT CODED    |XX/XX/XXXX
                     0 |//////////////////|         1234567  |        XX/XX/XXXX
-----------------------|------------------|------------------|------------------
c. DIRECTDPV           |                  |f. eLOT ASSIGNED  |XX/XX/XXXX
                     0 |                  |         1234567  |        XX/XX/XXXX
```

NOTE: USPS CASS regulations require Delivery Point Validation (DPV) processing for CASS certification. If you do not perform DPV processing, Finalist does not generate a USPS Form 3553 (CASS Summary Report).

CHAPTER 4

LACS^{Link} Option

This chapter describes the LACS^{Link} Option. The LACS^{Link} Option allows you to query the LACS^{Link} database to convert addresses and retrieve LACS^{Link} product name, version, and database date.

What is the LACS^{Link} Option?

The USPS LACS^{Link} database contains data on address conversions. An example is an address converted from a rural route/PO box address format to a number/street address format in order to receive 911 emergency response services. The USPS LACS^{Link} database provides mailers with a method for converting the old addresses. The USPS Change of Address (COA) database does not contain these addresses since these changes are actually conversions and not moves resulting in a change of address. Finalist will query the LACS^{Link} database to convert addresses as necessary. Through a PBFNInfo call, Finalist allows you to retrieve LACS^{Link} version and database date.

NOTE: USPS CASS regulations require LACS^{Link} processing for CASS certification and to generate the USPS Form 3553 (USPS CASS Summary Report).

Can I Use the LACS^{Link} Option?

The LACS^{Link} Option is available to all Finalist customers.

Why Should I Use the LACS^{Link} Option?

The USPS requires CASS-certified mailings to include LACS^{Link} processing. Some of the benefits available to Finalist customers through LACS^{Link} processing are described below.

- Ensures compliancy with USPS CASS regulations. LACS^{Link} processing is a USPS CASS regulations.
- Ensures preparedness for future USPS CASS regulations.
- Increased postal coding accuracy. Finalist processes all addresses that failed postal coding through LACS^{Link} processing.
- LACS^{Link} processing provides another method to ensure postal coding accuracy resulting in less undeliverable mail.
- More accurate and timely postal coding eliminates delivery delays.
- Improves delivery of targeted marketing mailings.

How Do I Install the LACS[Link] Option?

The LACS[Link] Option is installed as part of your standard Finalist installation. The distribution media sent to you in your release package contains all data needed to perform LACS[Link] processing. After installing Finalist, you must activate the LACS[Link] Option to perform LACS[Link] processing.

How Do I Activate LACS[Link] Processing?

To activate LACS[Link] processing, you will need to define the fields listed below.

- **LACS[Link]** — This field, in effect, turns on LACS[Link] processing.
- **LACS[Link] Processing** — This field determines the memory model used for LACS[Link] processing.
- **LACS[Link] Filepath** — Defines the location of LACS[Link] files.
- **LACS[Link] Key** — Ensures you are in compliance with your Pitney Bowes Software license agreement.

Methods for Activating LACS[Link] Processing

To activate LACS[Link] processing, use one of the following methods.

- **pbfn.cfg Configuration File** — If you prefer to configure your Finalist installation using global settings, you can define the LACS[Link] fields in your pbfn.cfg configuration file. For more information on the pbfn.cfg file, refer to your *Working With Finalist Guide*.
- **PBFNSetupDef Structure** — If you prefer to configure your Finalist installation using the Finalist structures, you can define the LACS[Link] fields in the PBFNSetupDef structure. For more information on the PBFNSetupDef structure, refer to your *Finalist Developer's Reference Guide*.
- **Workbench or Lookup Tool** — If you prefer to configure your Finalist installation using the Finalist Workbench or Lookup Tool GUI screens, you can define the LACS[Link] fields in the **Product Tab** on the *PBFN Config Setting* dialog box. For more information on the Finalist Workbench and Lookup Tool, refer to your *Working With Finalist Guide*.
- **Compatibility Interface (CI)** — If you prefer to configure your Finalist installation using the CI, you can define the LACS[Link] fields in the Finalist call area. For more information on the CI, refer to your *Finalist Developer's Reference Guide*.

Each method for activating LACS[Link] processing is described in the following sections.

Using pbfn.cfg to Activate LACS[Link] Processing

To activate LACS[Link] processing using the pbfn.cfg file, complete the following fields in your pbfn.cfg file.

Table 1: pbfn.cfg File — LACS[Link] Settings

pbfn.cfg Field	Description
LACSLink	Indicate whether to perform LACS[Link] processing. The USPS CASS regulations require LACS[Link] processing. If you do not perform LACS[Link] processing, Finalist does not generate a USPS Form 3553 (CASS Summary Report). Valid values are: • **OFF** — Do not perform LACS[Link] processing. Finalist does not generate the USPS Form 3553 (CASS Summary Report). • **ON** — Perform LACS[Link] processing. Finalist generates the USPS Form 3553 (CASS Summary Report). • blank — Defaults to **OFF**.
LACSLink Processing	Indicates the memory model for LACS[Link] processing: • **P** — Pico. Stores no data in memory. No tables or indexes are loaded. • **U** — Ultra-small. Stores no data in memory. Partial indexes are loaded. • **S** — Small • **M** — Medium • **L** — Large • **H** — Huge. Stores all data in memory. blank — Defaults to **S**.
LACSLink Filepath	Specify the LACS[Link] File path.
LACSLink Key	Enter the LACS[Link] activation key.

Using the PBFNSetupDef Structure to Activate LACS^{Link} Processing

To activate LACS^{Link} processing using the PBFNSetupDef structure, complete the following fields in the PBFNSetupDef structure.

Table 2: PBFNSetupDef Structure — LACS^{Link} Settings

PBFNSetupDef Field	Description
cAssignLACSLink	Indicate whether to perform LACS^{Link} processing. The USPS CASS regulations require LACS^{Link} processing. If you do not perform LACS^{Link} processing, Finalist does not generate a USPS Form 3553 (CASS Summary Report). Valid values are: • **OFF** — Do not perform LACS^{Link} processing. Finalist does not generate the USPS Form 3553 (CASS Summary Report). • **ON** — Perform LACS^{Link} processing. Finalist generates the USPS Form 3553 (CASS Summary Report). • blank — Defaults to **OFF**.
cLACSLinkProcessing	Indicates the memory model for LACS^{Link} processing: • **P** — Pico. Stores no data in memory. No tables or indexes are loaded. • **U** — Ultra-small. Stores no data in memory. Partial indexes are loaded. • **S** — Small • **M** — Medium • **L** — Large • **H** — Huge. Stores all data in memory. blank — Defaults to **S**.
cLACSLinkFilePath	Specify the LACS^{Link} File path.
cLACSLinkKey	Enter the LACS^{Link} activation key.

Using the Finalist Workbench to Activate LACS^{Link} Processing

To activate LACS^{Link} processing using the Finalist Workbench:

1. Launch the Finalist Workbench.
2. From the **Tools** menu, select **PBFN Setup**.

3. Select the **Product** tab on the *PBFN Config Setting* dialog box.

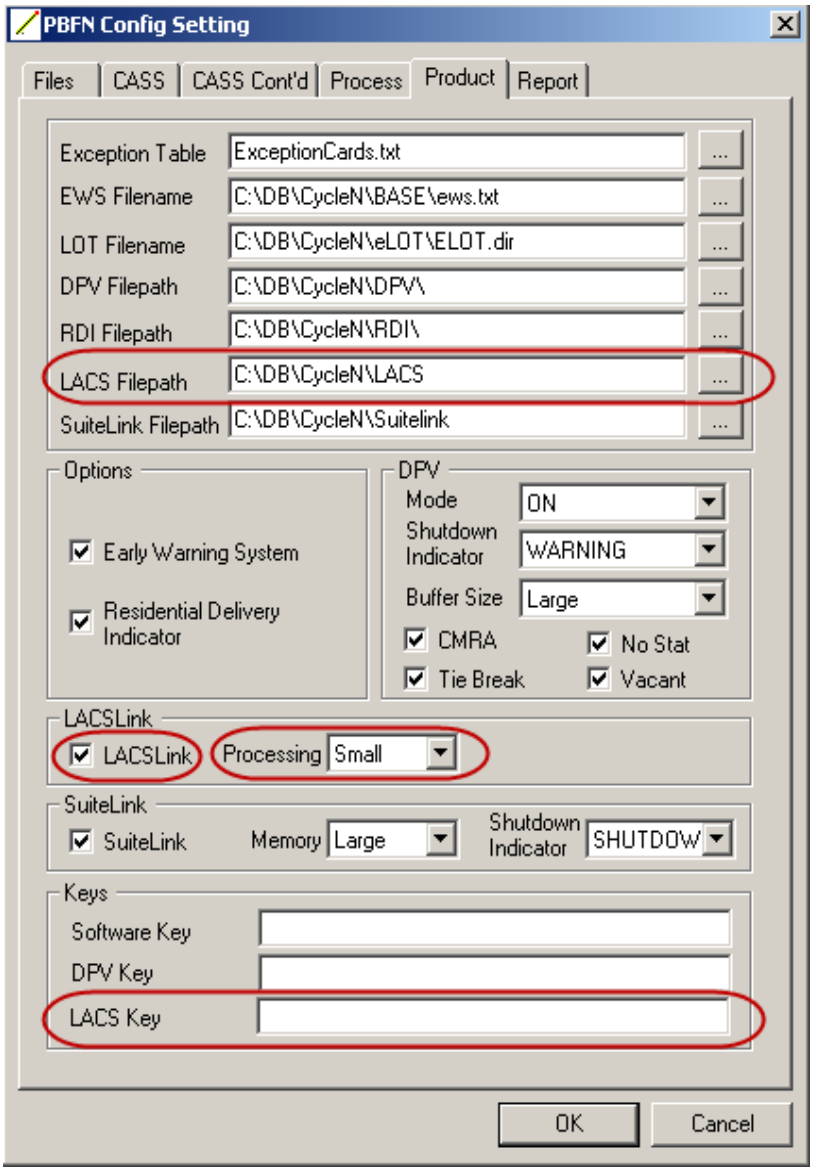

Figure 1: PBFN Config Setting Dialog Box — Product Tab

4. Complete the following fields on the **Product** Tab.

Table 3: PBFN Config Setting Dialog Box — Product Tab

Field	Description
LACS[Link] Filepath	Specify the LACS[Link] file path or click on the **Browse** button (located to the right of the field) to open a dialog box and point to the folder (path) that contains the databases.
LACS[Link]	Check the **LACS[Link]** box to perform LACS[Link] processing. The USPS CASS regulations require LACS[Link] processing. If you do not perform LACS[Link] processing, Finalist does not generate a USPS Form 3553 (CASS Summary Report).
LACS[Link] Processing	From the **LACS[Link] Processing** drop down list, select the LACS[Link] memory model for processing your file: • **Pico** — Pico. Stores no data in memory. No tables or indexes are loaded. • **Ultra** — Ultra-small. Stores no data in memory. Partial indexes are loaded. • **Small** — Small • **Medium** — Medium • **Large** — Large • **Huge** — Huge. Stores all data in memory. blank — Defaults to **Small**.
LACS Key	Enter the LACS[Link] activation key.

5. Click **OK** to save your settings.

Using the Compatibility Interface to Activate LACS[Link] Processing

To activate LACS[Link] processing using the CI, define the appropriate field for your platform in the Finalist call area.

Table 4: Compatibility Interface (CI) - LACS[Link] Settings

COBOL Field Name/ C Field Name	Field Description
FINAL-LLK-OPT caLLK	Determines whether Finalist performs LACS[Link] Option processing: • **N** — No LACS[Link] processing. • **Y** — LACS[Link] processing with Large memory model. • **U** — LACS[Link] processing with Ultra-small memory model. • **S** — LACS[Link] processing with Small memory model. • **M** — LACS[Link] processing with Medium memory model. • **L** — LACS[Link] processing with Large memory model. • **H** — LACS[Link] processing with Huge memory model. blank — Defaults to **S**.

How Does LACS[Link] Processing Work?

Finalist performs LACS[Link] processing in the situations described below.

1. Finalist codes an address that USPS data indicates has undergone a LACS conversion. In this case, the cLACS field in the PBFNProcessDataDef structure (or PBFNProcessDataAltDef structure for languages like COBOL that need to use character arrays in place of pointers for data fields) contains "L" indicating a LACS conversion has taken place. Finalist sends the record through LACS[Link] processing.

2. Finalist encounters a rural route address and cannot match the address to a Box number. Finalist codes the record to the rural route default level. Finalist sends any rural route record that cannot be matched to a Box through LACS[Link] processing.

3. Finalist sends any address that fails postal coding through LACS[Link] processing.

4. Finalist then processes the LACS[Link] returned address through the coding engine using the LACS[Link] returned address as the input address.

NOTE: For LACS converted addresses, the PBFNAddressInfoDef fields are based on the LACS returned address as input. The PBFNAddressInfoDef fields are not based on the original (pre-LACS) input address.

LACS[Link] Structures

This section provides an overview of the LACS[Link] structures. For detailed information on these structures refer to Chapter 2, Structures and Constants in your *Finalist Developer's Reference Guide*. Table 5 describes the LACS[Link] structures.

Table 5: LACS[Link] Structures (Part 1 of 2)

Structure	Description
PBFNRtnLACSStatsDef	You can pass the PBFNRtnLACSStatsDef structure on the PBFNStats or PBFNTerminate call to pass LACS[Link] processing statistics.
PBFNLACSSeedHdrDef	Use the PBFNLACSSeedHdrDef structure in on a PBFNStats call to return LACS[Link] False Positive violation header information.
USPSPBLACSHdrDef	The USPSPBLACSHdrDef structure returns the LACS[Link] False Positive violation header data required by the USPS for LACS[Link] processing in the USPS-required format.

Table 5: LACS^Link Structures (Part 2 of 2)

Structure	Description
PBFNLACSSeedDetDef	Use the PBFNLACSSeedDetDef structure on the PBFNProcess call to return the False Positive violation detail information for the USPS.
USPSPBLACSDetDef	The USPSPBLACSDetDef structure returns the LACS^Link False Positive violation detail data required by the USPS. For each False Positive (Seed) Table violation, a record is created in the USPS-required format.

Other Structures Containing LACS^Link Information

The structures listed below include data to facilitate LACS^Link processing. For detailed information on these structures, refer to your *Finalist Developer's Reference Guide*.

- PBFNAddressInfoDef
- PBFNIMSSetupDef
- PBFNInfoDef
- PBFNParsedAdrAltDef
- PBFNParsedAdrDef (includes LACS^Link return code)
- PBFNProcessDataAltDef
- PBFNProcessDataDef (includes LACS^Link return code)
- PBFNSetupDef (includes three LACS^Link setup parameters)
- PBFNStatsDef

LACS^Link Output

This following sections describe the output generated during LACS^Link processing.

LACS^Link Error Messages

The LACS^Link error messages are listed below. For more information, refer to the PBFNExtendedErrorDef structure.

Table 6: LACS^Link Option Error Messages

Error Message	Description
PCLERR_LACSINITERROR (20150)	System errors at LACS^Link
PCLERR_LACSINITFAIL (20151)	Occurred during LACS^Link initialization
PCLERR_LACSVIOLATION (20152)	LACS^Link seed violation
PCLERR_LACSDBEXPIRED (20153)	LACS^Link database expired
PCLERR_LACSKEYERROR (20154)	Invalid LACS^Link key
PCLERR_LACSTERMFAIL (20155)	LACS^Link termination error

LACS^Link Return Codes

If LACS^Link processing changes an address, the cLACSRtnCode field in the PBFNProcessDataDef structure (or PBFNProcessDataAltDef for languages like COBOL that need to use character arrays in place of pointers for data fields) contains one of the return codes listed below.

Table 7: LACS^Link Option Return Codes

Return Code	Description
A	LACS^Link processing successful. Record matched through LACS^Link processing.
00	LACS^Link processing failed. No matching record found during LACS^Link processing.
09	LACS^Link processing matched the input address to an older highrise default address. The address has been converted. However, rather than provide an imprecise address, LACS^Link processing does not provide a new address.
14	LACS^Link processing failed. Match found during LACS^Link processing but conversion did not occur due to other USPS regulations.
92	LACS^Link processing successful. Record matched through LACS^Link processing. Unit number dropped on input.

LACS[Link] Information on Finalist Batch Report

The Finalist Batch Report provides statistics for LACS[Link] processing. The Finalist Batch Report displays the LACS[Link] settings from the pbfn.cfg configuration file or the PBFNSetupDef structure and the LACS[Link] processing counts. For more information on the Finalist Batch Report, refer to your *Working With Finalist Guide*.

LACS[Link] Information on USPS Form 3553 (CASS Summary Report)

The Qualitative Statistical Summary (QSS) section of the USPS Form 3553 (CASS Summary Report) displays as shown below if you are processing with the LACS[Link] Option.

```
================================================================================
== E. QUALITATIVE STATISTICAL SUMMARY (QSS) ====================================
HR DEFAULT |  HR EXACT | RR DFLT  | RR EXACT  |  LACSLINK | EWS      | SUITELINK
       5830|      22554|         0|       1922|       1023|        28|         68
================================================================================
PS Form 3553, XXXXXXXX XXXX
```

NOTE: USPS CASS regulations require LACS[Link] processing. If you do not perform LACS[Link] processing, Finalist does not generate a USPS Form 3553 (CASS Summary Report).

CHAPTER 5

Resolving False Positive Violations

This chapter provides information for resolving DPV and LACS^{Link} False Positive violations.

What is a False-Positive Violation?

The USPS has put security measures in place to ensure mailers using DPV and LACS^{Link} processing do not use these applications to generate mailing lists. False Positive (Seed) records are artificially manufactured addresses provided as part of the DPV and LACS^{Link} options. If the USPS identifies a mailer as repetitively generating False Positive (Seed) violations, the USPS may direct Pitney Bowes Software to invalidate their license. Towards that end, the USPS monitors addresses that generate a False Positive result. The USPS requires Pitney Bowes Software to report any organization generating a False Positive result during DPV and/or LACS^{Link} processing. If you generate a False Positive result, Finalist generates an error message indicating a False Positive (Seed) violation.

For the job that encountered the False Positive (Seed) violation, the CASS statement will be provided showing the number of records that were confirmed up until the point of the False Positive (Seed) violation. No records will be confirmed after the False Positive (Seed) violation occurred.

For any job set to perform CASS processing and submitted after a the False Positive (Seed) violation occurs, Finalist generates an initialization error and stops processing.

How Do I Know I Have Hit a Seed Violation

This section provides information for determining whether you have hit a seed violation.

Batch Processing

If you encounter a seed violation using the batch application, the following occurs:

1. The function (DPV or LACS^{Link}) generating the seed violation stops processing.

2. The Finalist job continues to run to completion if the DPV Shutdown Indicator is not set to "S". Otherwise, a seed violation will stop the Finalist job.

3. An error message is written to the log.

Table 1: Batch Processing Seed Violation Error Messages

Process	Seed Violation Message
DPV	20125 DPV Processing Error. Seed violation encountered. Call PB-DMT tech support Case 8 DPV 8.
LACS^{Link}	20152 LACS Processing Error. Seed Violation encountered. Call PB-DMT tech support.

4. Finalist generates the USPS Form 3553 (CASS Summary Report) for the job that encountered the False Positive (Seed) violation. It will be reflective of the records that were DPV/LACS^{Link} processed before the False Positive (Seed) violation.

5. Finalist writes the offending record to the SEEDLOG file. For Mainframe environments, your JCL includes a DD statement for SEEDLOG. The seed violation record is written to the location and filename assigned in this DD statement. For Windows and Unix environments, Finalist creates a "seedlog.txt" file in your /bin directory containing the seed violation.

6. In addition to producing a Seed Log, the output file indicates seeds by:

    a. If a DPV seed violation is encountered, the output file contains a "Y" for the DPV Flags False Positive indicator in position 3 of the three-byte output field oDpvFlags.

    b. If a LACS^{Link} violation is encountered, the output LACS^{Link} Seed Detail field oLLKSD is populated with a "Y" if a False Positive (Seed) violation was encountered. oLLKSD is not available to be posted out with the Workbench for Windows.

7. The function (DPV or LACS^{Link}) generating the seed violation cannot process subsequent jobs until a reactivation key is applied. Any CASS job submitted after the False Positive (Seed) violation will encounter an initialization error.

Calling Finalist

The Finalist engine (PBFN.dll) automatically creates the seedlog file. Calling driver programs are no longer responsible for creating the seedlog file:

- **Windows and Unix** — The seedlog file generated is seedlog.txt.
- **MVS** — Define the following in your JCL:

```
//SEEDLOG  DD  DSN=hlq.SEEDLOG,
//         DISP=(MOD,CATLG),
//         DCB=(LRECL=180,BLKSIZE=0,RECFM=FB),
//         SPACE=(TRK,(1,1),RLSE)
```

What to do When You Encounter a Seed

For all forms of processing with Finalist, if you encounter a seed, you must report the seed violation to the USPS. We have made available, via our Support site at http://www.g1.com/support, a method for reporting all seed violations to the USPS and for obtaining re-activation keys and/or updated security files.

Reporting Seed Violations

This section provides information on the requirements and steps for reporting DPV and LACS^{Link} seed violations.

The Finalist engine (PBFN.dll) automatically creates the Seed Log file. Driver programs are not responsible for creating the Seed Log file. This applies to custom batch drivers calling FINALIST on the mainframe.

- If you perform Finalist processing using the Native or Compatibility Interface batch mode, Finalist generates the Seed Log and writes the Seed Log out to an output file that is defined using the DD SEEDLOG JCL statement on the Mainframe platform:

```
//SEEDLOG  DD  DSN=hlq.SEEDLOG,
//         DISP=(MOD,CATLG),
//         DCB=(LRECL=180,BLKSIZE=0,RECFM=FB),
//         SPACE=(TRK,(1,1),RLSE)
```

- For Windows and Unix, Finalist generates a "seedlog.txt" file automatically and places it in the /bin directory. The same Seed Log output file is used for both DPV and LACS^{Link} seed violations.

The formats for the Header and Detail records are identical for DPV and LACS^{Link}.

Header Record

The header record should be created in the following format.

Table 2: Header Record

Position	Length	Description	Format
1-40	40	Mailer's company name	Alphanumeric
41-98	58	Mailer's address line	Alphanumeric
99-126	28	Mailer's city name	Alphanumeric
127-128	2	Mailer's state abbreviation	Alphabetical
129-137	9	Mailer's nine-digit ZIP Code	Numeric
138-146	9	Total records processed	Numeric
147-155	9	Total records DPV matched	Numeric
156-164	9	Percent match rate to DSF	Numeric
165-173	9	Percent match rate to ZIP + 4	Numeric
174-178	5	Number of ZIP Codes on file	Numeric
179-180	2	Number of false positives	Numeric

Detail Record

The detail record should be organized in the following format.

Table 3: Detail Record (Part 1 of 2)

Position	Length	Description	Format
1-2	2	Street pre-directional	Alphanumeric
3-30	28	Street name	Alphanumeric
31-34	4	Street suffix abbreviation	Alphanumeric
35-36	2	Street post-directional	Alphanumeric
37-46	10	Address primary number	Alphanumeric
47-50	4	Address secondary abbreviation	Alphanumeric
51-58	8	Address secondary number	Numeric
59-63	5	Matched ZIP Code	Numeric

Table 3: Detail Record (Part 2 of 2)

Position	Length	Description	Format
64-67	4	Matched ZIP + 4	Numeric
68-180	113	Filler	

Accessing the Seed Violation Reporting/Key Support Site

After acquiring your Seed Log and accessing the Support site to report the False Positive (Seed) violation, you will be provided with a reactivation key to be applied to restart the function in which the violation occurred. Follow these steps to access the False Positive (Seed) Violation Reporting/Key area of the Support site:

1. Login to the Pitney Bowes Software Support site, http://www.g1.com/Support.
2. Go to "My Products".
3. Select "Finalist".
4. Select the platform on which you process with Finalist.
5. Under the Database section, you will see listings for:
    a. DPV (for the purpose of keys, this link covers DPV Split File, Full File and Flat File DPV implementations)
    b. LACS^{Link}
6. Under the License column of the Database table, click "View Available Downloads" for the database of the function generating the seed violation.
7. You will be prompted for the information to enter for the affected function and the type of processing that you were performing at the time of the violation.

Obtaining a Re-Activation Key or Security File for Batch Jobs

To obtain a re-activation key or security file, you will need to select Batch, via the license form. If you encountered a DPV or LACS^{Link} Seed Violation while processing a Finalist batch job, you are required to indicate whether you are processing with the Compatibility Interface or Native (Configuration File).

- If you select Native for the mainframe platform, you must enter the current DPV or LACS^{Link} key, shown in your configuration file (pbfn.cfg) and attach the SEEDLOG file downloaded from your mainframe as ASCII to be issued a re-activation key. After entering the DPV/LACS^{Link} key and the Seed File, you can select the Download License button to acquire your DPV or LACS^{Link} re-activation key.
- If you select Compatibility Interface for the mainframe platform, you will only need to attach the Seed File and click the Download License button to acquire your DPV or LACS^{Link} re-activation security file.
- For the Windows and Unix platforms, you must enter your current DPV or LACS^{Link} key, shown in your configuration file (pbfn.cfg) and attach the "seedlog.txt" file to be issued a re-activation key. After entering the key and attaching the Seed File, you can select the Download License button to acquire your DPV or LACS^{Link} re-activation key.

Installing the Re-Activation Key or Security File for Batch Jobs

Follow the instructions below for your platform to install your re-activation key or security file.

- If you select Native for the mainframe platform, to install the re-activation key, you can copy and paste the new key or type over the existing key in your existing configuration file (pbfn.cfg).
- If you selected the Compatibility Interface for the mainframe platform, to install the re-activation security file, you must FTP this file as binary, replacing your existing DPVSUD or LLKSUD file accordingly. The DPVSUD and LLKSUD files are loaded at the time of the software installation process. The format of both of these security files is: RECFM=F, LRECL=7 and BLKSIZE=7.
- For the Windows and Unix platforms, to install the re-activation key, you can copy and paste the new key or type over the existing key in your existing configuration file (pbfn.cfg).

Finalist Structures Containing False Positive Violation Information

The structures listed below provide information on False Positive results generated during DPV and/or LACS^{Link} processing. For more information on these structures, refer to your *Finalist Developer's Reference Guide*.

Table 4: Structures Containing False Positive Violation Information

Structure	Description
USPSDPVHdrDef	Returns the header data required by the USPS for DPV False Positive (Seed) Table violations in the USPS-required format.
USPSDetailDef	Returns the detail data required by the USPS for each record creating a DPV Option False Positive (Seed) Table violation in the USPS-required format.
USPSPBLACSDetDef	Returns the LACS^{Link} violation detail data required by the USPS. Finalist creates a record in the USPS-required format for each False Positive (Seed) Table violation.
USPSPBLACSHdrDef	Returns the LACS^{Link} False Positive violation header data required by the USPS for LACS^{Link} processing in the USPS-required format.

CHAPTER 6

Line of Travel (eLOT) Option

This chapter describes the Finalist Line of Travel (eLOT) Option. You can use the Line of Travel (eLOT) Option to add eLOT codes to mailings.

What is the Line of Travel (eLOT) Option?

Line of travel sequence is an option for mailers who prepare carrier route mailings other than high-density/125-piece or saturation mailings. eLOT sequencing is required for Basic Enhanced Carrier Route Standard Mail except automation-compatible, letter-size pieces. eLOT sequence is not an exact walk sequence but a sequence of ZIP + 4 Codes arranged in the order that the route is served by a carrier. First the ZIP + 4 groups are sequenced. Then the addresses within each group are identified as being in ascending or descending order.

Finalist releases include a monthly eLOT database. The eLOT database ensures that Enhanced Carrier Route mailings are sorted much closer to the actual delivery sequence. The Finalist database and eLOT database must be in synch (i.e., September eLOT data must be processed with a September Finalist database). The PBFNInfoDef structure and Batch Report contain the eLOT database version number, maintenance dates, and expiration dates. If the Finalist database and the eLOT database are not in synch, there may be ZIP + 4 Codes for which eLOT numbers cannot be assigned. The ZIP Code, ZIP + 4 code, carrier route code, and the delivery point of an address must be provided to assign a eLOT code.

Can I Use the Line of Travel (eLOT) Option?

The eLOT Option is available to all Finalist customers.

Why Should I Use the Line of Travel (eLOT) Option?

Some of the benefits available to Finalist customers through eLOT processing are described below.

- The USPS requires eLOT sequencing for Basic Enhanced Carrier Route Standard Mail except automation-compatible, letter-size pieces.
- The eLOT Option ensures that Enhanced Carrier Route mailings are sorted much closer to the actual delivery sequence.

How Do I Install the Line of Travel (eLOT) Option?

The eLOT Option is installed as part of your standard Finalist installation. The distribution media sent to you in your release package contains all data needed to perform eLOT processing. After installing Finalist, you must activate the eLOT Option to perform eLOT processing.

How Do I Activate Line of Travel (eLOT) Processing?

To activate eLOT processing, you will need to define the fields listed below.

- **LOT Filename** — Use this field to specify the name of the eLOT File.
- **Assign LOT** — Use this field to indicate whether to perform eLOT processing. To return the proper eLOT code field in the PBFNProcessDataDef structure, set this parameter to ON. If you do not want to retrieve the eLOT value for successfully-coded addresses, set this parameter to OFF. The default value for this field is OFF.
- **Process LOT Only** — Use this field to indicate whether to perform Finalist processing with eLOT processing or to perform eLOT processing only.

Methods for Activating eLOT Processing

To activate eLOT processing, use one of the following methods.

- **pbfn.cfg Configuration File** — If you prefer to configure your Finalist installation using global settings, you can define the eLOT fields in your pbfn.cfg configuration file. For more information on the pbfn.cfg file, refer to your *Working With Finalist Guide*.
- **PBFNSetupDef Structure** — If you prefer to configure your Finalist installation using the Finalist structures, you can define the eLOT fields in the PBFNSetupDef structure. For more information on the PBFNSetupDef structure, refer to your *Finalist Developer's Reference Guide*.
- **Workbench or Lookup Tool** — If you prefer to configure your Finalist installation using the Finalist Workbench or Lookup Tool GUI screens, you can define the eLOT fields using the **Product Tab** and the **Process Tab** on the *PBFN Config Setting* dialog box. For more information on the Finalist Workbench and Lookup Tool, refer to your *Working With Finalist Guide*.
- **Compatibility Interface (CI)** — If you prefer to configure your Finalist installation using the CI, you can define the eLOT fields in the Finalist call area. For more information on the CI, refer to your *Finalist Developer's Reference Guide*.

Each method for activating eLOT processing is described in the following sections.

Using pbfn.cfg to Activate eLOT Processing

To activate eLOT processing using the pbfn.cfg file, complete the following fields in your pbfn.cfg file.

Table 1: pbfn.cfg File — eLOT Settings

pbfn.cfg Field	Description
LOT Filename	Specify the Line of Travel (eLOT) file name and path.
Assign LOT	Indicate whether to assign Line of Travel (eLOT) codes: • **OFF** — Do not assign Line of Travel (eLOT) codes. • **ON** — Assign Line of Travel (eLOT) codes. • blank — Defaults to **OFF**.
Process LOT Only	Indicate whether to perform only eLOT processing: • **OFF** — Perform normal Finalist processing. • **ON** — Perform eLOT processing only. **If you specify "ON" for this field, Finalist does not perform address cleansing.** A fully-coded address record will be passed to the engine. Finalist performs an eLOT lookup and returns the eLOT code. • blank — Defaults to **OFF**.

Using the PBFNSetupDef Structure to Activate eLOT Processing

To activate eLOT processing using the PBFNSetupDef structure, complete the following fields in the PBFNSetupDef structure.

Table 2: PBFNSetupDef Structure — eLOT Settings

PBFNSetupDef Field	Description
cLOTFileName	Specify the name of the Line of Travel (LOT) file.
cAssignLOT	Indicate whether to assign Line of Travel (LOT) codes: • **OFF** — Do not assign Line of Travel (LOT) codes. • **ON** — Assign Line of Travel (LOT) codes. • blank — Defaults to **OFF**.
cProcessLOTOnly	Indicate whether to perform only eLOT processing: • **OFF** — Perform normal Finalist processing. • **ON** — Perform eLOT processing only. **If you specify "ON" for this field, Finalist does not perform address cleansing.** A fully-coded address record will be passed to the engine. Finalist performs an eLOT lookup and returns the eLOT code. • blank — Defaults to **OFF**.

Using the Finalist Workbench to Activate eLOT Processing

To activate eLOT processing using the Finalist Workbench:

1. Launch the Finalist Workbench.
2. From the **Tools** menu, select **PBFN Setup**.
3. Select the **Product** tab on the *PBFN Config Setting* dialog box.

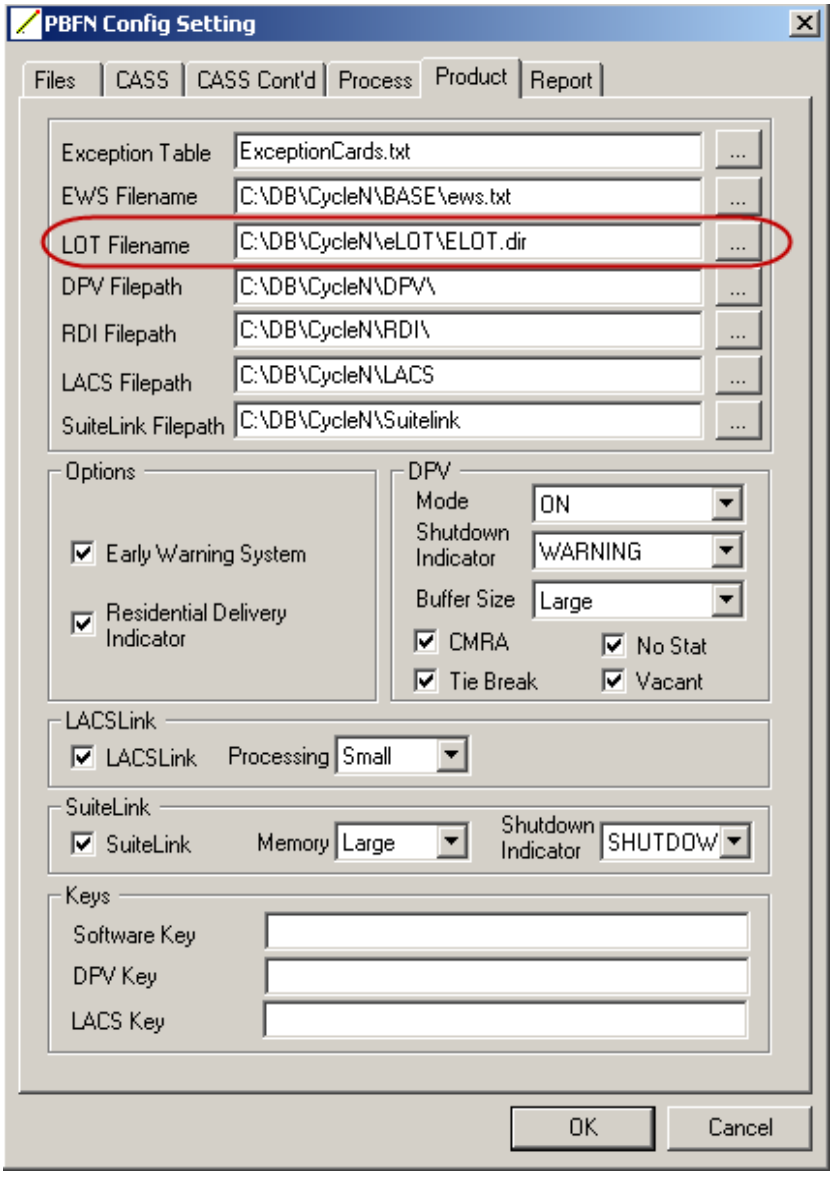

Figure 1: PBFN Config Setting Dialog Box — Product Tab

4. Complete the following fields on the **Product** Tab.

Table 3: PBFN Config Setting Dialog Box — Product Tab

Field	Description
LOT Filename	Specify the Line of Travel (LOT) file name and path or click on the **Browse** button (located to the right of the field) to open a dialog box and select the path and file name from a list of existing files.

5. Select the **Process** tab on the *PBFN Config Setting* dialog box.

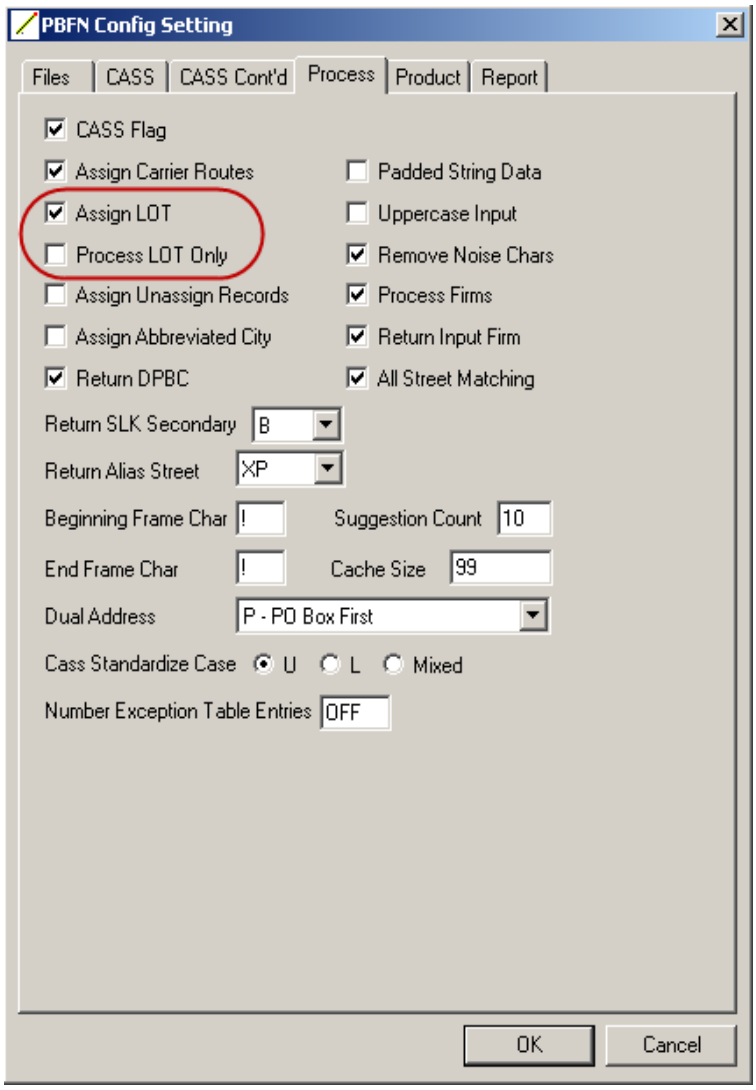

Figure 2: PBFN Config Setting Dialog Box — Process Tab

6. Complete the following fields on the **Process** tab.

Table 4: PBFN Config Setting Dialog Box — Process Tab

Field	Description
Assign LOT	Indicate whether to assign Line of Travel (LOT) codes: • **OFF** — Do not assign Line of Travel (LOT) codes. • **ON** — Assign Line of Travel (LOT) codes. • blank — Defaults to **OFF**.
Process LOT Only	Indicate whether to perform only eLOT processing: • **OFF** — Perform normal Finalist processing. • **ON** — Perform eLOT processing only. **If you specify "ON" for this field, Finalist does not perform address cleansing.** A fully-coded address record will be passed to the engine. Finalist performs an eLOT lookup and returns the eLOT code. • blank — Defaults to **OFF**.

7. Click **OK** to save your settings.

Using the Compatibility Interface (CI) to Activate eLOT Processing

To activate eLOT processing using the CI, define the appropriate field for your platform in the Finalist call area.

Table 5: Compatibility Interface (CI) - eLOT Settings

COBOL Field Name/ C Field Name	Field Description
FINAL-LOT-OPT caLOT	Determines whether Finalist performs eLOT processing. If this field contains a "Y", the system performs eLOT processing.

The option to perform eLOT processing only is not supported in the CI.

Assigning Line of Travel (eLOT) Codes

Finalist provides two methods for assigning eLOT codes when calling Finalist (Native).

- **Single-pass processing** — eLOT codes are assigned during an address correction run.
- **Two-pass processing** — eLOT codes are assigned independently in an eLOT-only assignment run.

Each of these methods is described in detail next.

Assigning eLOT Codes in a Single-Pass Process

To assign eLOT codes, you must provide the LOT File name and activate the assignment indicator in the pbfn.cfg file or the PBFNSetupDef structure. For detailed information, refer to the section "How Do I Activate Line of Travel (eLOT) Processing?" on page 67.

During the PBFNInit, if you have not defined the name of the eLOT File or the open has failed, the PBFN_NOLOT return code is issued. This return code indicates that the Zip4us.dir and the City.dir Files were successfully opened and address processing can continue. However, eLOT codes will not be assigned.

If you did not set the cLOTCode [LOT_LEN] or the LOT pointer is null in the PBFNProcessDataDef structure, eLOT codes will not be assigned and returned for that record. If a record is successfully coded, but the call to the eLOT database failed to return a valid eLOT code, the PBFNProcess call will return PBFN_SUCCESS and issue an error code. For detailed information on error codes, refer to "Line of Travel (eLOT) Output" on page 73.

Assigning eLOT Codes in a Two-Pass Process

To assign eLOT codes only, the addresses to code must contain valid ZIP Codes, ZIP + 4 Codes, carrier route codes, and delivery point barcode values. To assign eLOT codes, you must provide the eLOT File name and activate the assignment indicator in the pbfn.cfg file or the PBFNSetupDef structure. For detailed information, refer to the section "How Do I Activate Line of Travel (eLOT) Processing?" on page 67. Only the eLOT File is required for two-pass processing. To assign eLOT codes in a two-pass process, follow the steps described next.

1. Initialize the engine to use eLOT File only as shown next.

```
long PBFN_API PBFNInit(char *initType,
                       PBFNExtendedErrorDef *ExtError,
                       PBFNSetupDef *pSetup,
                       void *Reserved1,
                       void *Reserved2,
                       void *Reserved3,
                       void *Reserved4,
                       void *Reserved5,
                       void *Reserved6);
```

   a. **pSetup** — Pointer to a data structure (by reference) of type PBFNSetupDef that contains setup information.

   b. **initType** — Defined as PBFN_INIT_LOTONLY.

2. To pass addresses into the engine for coding, make the PBFNProcess call. Make sure the key values of cZip, cZip4, cCrRte, and cDelPoint are filled in the PBFNProcessDataDef structure. The cLOT and cLOTCode[LOT_LEN] fields must be allocated in the return structure for LOT data to be returned. Address correction will not occur during this call.

3. The PBFNStats API and the report APIs will only contain valid data for the LOT fields.

4. Call PBFNTerminate to internally deallocate only the eLOT-required memory and files.

Line of Travel (eLOT) Output

The following sections describe the output generated during eLOT processing.

eLOT Return Value

PBFNInit may include the following return value.

Table 6: eLot Option Return Value

Error Message	Description
PBFN_NOLOT	The eLOT File has not been defined or the open has failed. The Zip4us.dir and the City.dir Files were successfully opened. Address processing can continue. eLOT codes will not be assigned. This return code is only returned on one-pass processing. The value for this return code is 8.

eLOT Error Message

PBFNProcess may include the following error message.

Table 7: eLot Option Error Message

Error Message	Description
4461 - eLOT assignment has failed	Address coded successfully but eLOT code was not assigned.

eLOT Information on the Finalist Batch Report

The Finalist Batch Report displays the eLOT settings from the pbfn.cfg configuration file or the PBFNSetupDef structure and the eLOT processing counts. For detailed information on the Finalist Batch Report, refer to your *Working With Finalist Guide*.

eLOT Information on the USPS Form 3553 (CASS Summary Report)

The C. Output section of the USPS Form 3553 (CASS Summary Report) shows the number of addresses assigned eLOT codes. For detailed information the USPS Form 3553 (CASS Summary Report), refer to your *Working With Finalist Guide*.

```
== C. OUTPUT ========================================================================
OUTPUT RATING          |VALIDATION PERIOD |OUTPUT RATING     |VALIDATION PERIOD
            TOTAL CODED|FROM    TO        |       TOTAL CODED|FROM    TO
-----------------------|------------------|------------------|------------------
a. ZIP+4/DPV CONFIRMED |XX/XX/XXXX        |d. 5-DIGIT CODED  |XX/XX/XXXX
                1234567 |        XX/XX/XXXX|         1234567  |        XX/XX/XXXX
-----------------------|------------------|------------------|------------------
b. Z4CHANGE PROCESSED  |//////////////////|e. CR RT CODED    |XX/XX/XXXX
                     0 |//////////////////|         1234567  |        XX/XX/XXXX
-----------------------|------------------|------------------|------------------
c. DIRECTDPV           |                  |f. eLOT ASSIGNED  |XX/XX/XXXX
                     0 |                  |         1234567  |        XX/XX/XXXX
```

CHAPTER 7

Early Warning System (EWS) Option

This chapter describes the Finalist Early Warning System (EWS) Option. You can use the Early Warning System (EWS) Option to verify input addresses not found in the current ZIP + 4 File against the USPS EWS File.

What is the Early Warning System (EWS) Option?

The USPS postal database may not contain the most current address information. New address information that is in use, but not yet available on the ZIP + 4 File, can now be found as part of the CASS Department's Early Warning System (EWS). These new or changed addresses can be found at the USPS RIBBS web site https://ribbs.usps.gov/cassmass/documents/tech_guides/. The USPS has scheduled the EWS File for update on a weekly basis. The EWS File on the USPS site is available for users to download. Pitney Bowes will send a monthly update of the EWS File with the database updates.

USPS CASS regulations require all CASS-certified software to be able to read the USPS Early Warning System (EWS) File. The Finalist Early Warning System (EWS) Option verifies input addresses that are not found in the current ZIP + 4 File against the USPS EWS File. If an input address is found in the EWS File, the input address is not matched to any similar addresses in the current ZIP + 4 File. Instead, the input address fails and is not coded until the ZIP + 4 File is updated with the correct address from the USPS EWS File.

For example, your input file contains the address "100 S. Bonnie Ct". Finalist does not find an exact match on the current ZIP + 4 File. The current ZIP + 4 File contains the address range 100 - 198 Bonnie Ave. Finalist would normally code the address as "100 Bonnie Ave". If Finalist finds values for "S Bonnie Ct" for the input ZIP Code in the EWS File, CASS regulations require Finalist to fail the address and not update the address until the USPS adds the valid address to the ZIP + 4 File.

Can I Use the Early Warning System (EWS) Option?

The EWS Option is available to all Finalist customers.

Why Should I Use the Early Warning System (EWS) Option?

The USPS CASS regulations require all CASS-certified software to be able to read the USPS Early Warning System (EWS) File. Some of the benefits available to Finalist customers through EWS processing are described below.

- Allows you to verify your input addresses that are not found in the current ZIP + 4 File against the USPS EWS File.
- Ensures that input addresses not found in the ZIP + 4 file are not incorrectly coded or deleted.
- Provides access to addresses that will be added to the ZIP + 4 file in the next few months.

How Do I Install the EWS Option?

The EWS Option is installed as part of your standard Finalist installation. The distribution media sent to you in your release package contains all data needed to perform EWS processing. After installing Finalist, you must activate the EWS Option to perform EWS processing.

How Do I Activate EWS Processing?

To activate EWS processing, you will need to define the fields listed below.

- **EWS Filename** — Define the name of the EWS file.
- **Early Warning System** — This field indicates whether to perform Early Warning System (EWS) processing.

Methods for Activating EWS Processing

To activate EWS processing, use one of the following methods.

- **pbfn.cfg Configuration File** — If you prefer to configure your Finalist installation using global settings, you can define the EWS fields in your pbfn.cfg configuration file. For more information on the pbfn.cfg file, refer to your *Working With Finalist Guide*.
- **PBFNSetupDef Structure** — If you prefer to configure your Finalist installation using the Finalist structures, you can define the EWS fields in the PBFNSetupDef structure. For more information on the PBFNSetupDef structure, refer to your *Finalist Developer's Reference Guide*.
- **Workbench or Lookup Tool** — If you prefer to configure your Finalist installation using the Finalist Workbench or Lookup Tool GUI screens, you can define the EWS fields in the **Product Tab** on the *PBFN Config Setting* dialog box. For more information on the Finalist Workbench and Lookup Tool, refer to your *Working With Finalist Guide*.
- **Compatibility Interface (CI)** — If you prefer to configure your Finalist installation using the CI, you can define the EWS fields in the Finalist call area. For more information on the CI, refer to your *Finalist Developer's Reference Guide*.

Each method for activating EWS processing is described in the following sections.

Using pbfn.cfg to Activate EWS Processing

To activate EWS processing using the pbfn.cfg file, complete the following fields in your pbfn.cfg file.

Table 1: pbfn.cfg File — EWS Settings

pbfn.cfg Field	Description
EWS Filename	Specify the Early Warning System (EWS) file name and path.
Early Warning System	Indicate whether to perform Early Warning System (EWS) processing: • **OFF** — Do not perform Early Warning System (EWS) processing. • **ON** — Perform Early Warning System (EWS) processing. • blank — Defaults to **OFF**.

Using the PBFNSetupDef Structure to Activate EWS Processing

To activate EWS processing using the PBFNSetupDef structure, complete the following fields in the PBFNSetupDef structure.

Table 2: PBFNSetupDef Structure — EWS Settings

PBFNSetupDef Field	Description
cEWSFileName	Specify the Early Warning System (EWS) file name and path.
cAssignEWS	Indicate whether to perform Early Warning System (EWS) processing: • **OFF** — Do not perform Early Warning System (EWS) processing. • **ON** — Perform Early Warning System (EWS) processing. • blank — Defaults to **OFF**.

Using the Finalist Workbench to Activate EWS Processing

To activate EWS processing using the Finalist Workbench:

1. Launch the Finalist Workbench.
2. From the **Tools** menu, select **PBFN Setup**.

3. Select the **Product** tab on the *PBFN Config Setting* dialog box.

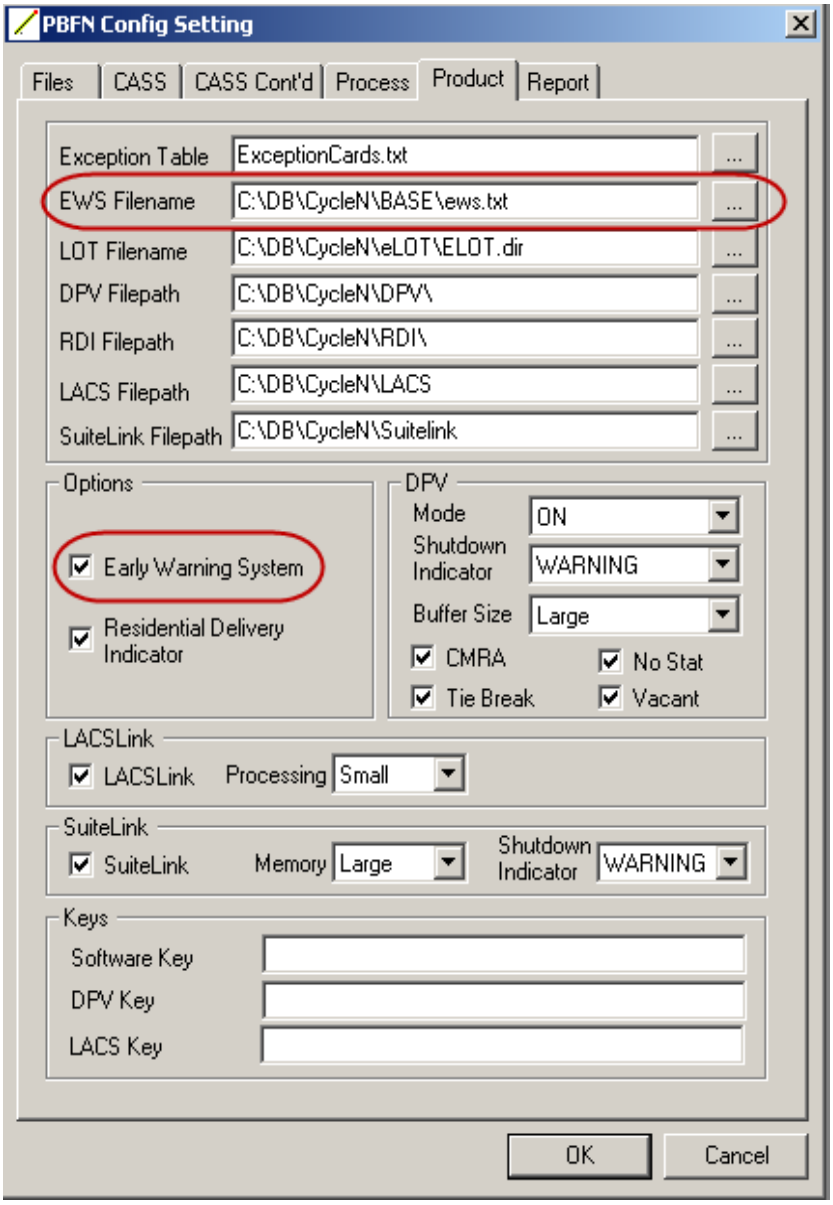

Figure 1: PBFN Config Setting Dialog Box — Product Tab

4. Complete the following fields on the **Product** tab.

Table 3: PBFN Config Setting Dialog Box — Product Tab

Field	Description
EWS Filename	Specify the Early Warning System (EWS) file name and path or click on the **Browse** button (located to the right of the field) to open a dialog box and select the path and file name from a list of existing files.
Early Warning System	Check the Early Warning System box to perform Early Warning System (EWS) processing.

5. Click **OK** to save your settings.

Using the Compatibility Interface (CI) to Activate EWS Processing

To activate EWS processing using the CI, define the appropriate field for your platform in the Finalist call area.

Table 4: Compatibility Interface (CI) - EWS Settings

COBOL Field Name/ C Field Name	Field Description
FINAL-EWS-OPT caEWS	Determines whether Finalist performs Early Warning System (EWS) processing. If this field contains a "Y", Finalist performs EWS processing.

How Does EWS Processing Work?

Finalist performs EWS processing as described below.

1. During processing of your input file, Finalist identifies an address that is not an exact match to the current ZIP + 4 file.
2. Finalist compares the address to the EWS File.
3. If Finalist finds the inexact address on the EWS File, per CASS regulations, Finalist fails the address and does not update the address until the USPS adds the valid address to the ZIP + 4 File.
4. The number of failed EWS addresses displays on the Finalist Batch Report and the USPS Form 3553 (CASS Summary Report).

5. On the z/OS platform, there are two methods available for working with EWS — the CBEWS file and the PBFNEWS module.

   - **CBEWS** — CBEWS is a VSAM file that contains all of the EWS records from the USPS. It should be populated when the USPS releases EWS data (unless you always populate the PBFNEWS module, see below).
   - **PBFNEWS** — PBFNEWS is a module that may or may not contain the same EWS records from the USPS. The default configuration shipped with Finalist has no EWS records in the PBFNEWS module. If you run the LOADEWS JCL (in FNSOURCE), LOADEWS populates the PBFNEWS module.
     - If PBFNEWS is NOT populated when Finalist runs, Finalist reads CBEWS and stores the data internally.
     - If PBFNEWS is populated when Finalist runs, Finalist does not need to read the CBEWS file. This method saves time and I/O counts, especially critical in an online (CICS or IMS) environment.
     - If you are ALWAYS going to populate the PBFNEWS module (LOADEWS), you do not need to load the data into CBEWS. Also, in this case, the CBEWS DD does not need to be part of your JCL stream.

       NOTE: This method will provide the best performance of EWS processing.
     - If you do not want to run LOADEWS, then you must load data into the CBEWS file.

Structures Containing EWS Information

The structures listed below include data to facilitate EWS processing. For detailed information on these structures, refer to your *Finalist Developer's Reference Guide*.

- PBFN3553Def
- PBFNAddressInfoDef
- PBFNIMSSetupDef
- PBFNSetupDef (includes three EWS setup parameters)
- PBFNStatsDef

EWS Output

This following sections describe the output generated during EWS processing.

EWS Error Messages

The EWS error messages are listed below. For more information, refer to the PBFNExtendedErrorDef structure.

Table 5: EWS Option Error Messages

Error Message	Description
4460: EWS Failure. Address found on EWS table.	The indicated address was found on the EWS table.

EWS Information on Finalist Batch Report

The Finalist Batch Report displays the EWS settings from the pbfn.cfg configuration file or the PBFNSetupDef structure and the EWS processing counts. For detailed information on the Finalist Batch Report, refer to your *Working With Finalist Guide*.

EWS Information on USPS Form 3553 (CASS Summary Report)

The Qualitative Statistical Summary (QSS) section of the USPS Form 3553 (CASS Summary Report) displays as shown below if you are processing with the EWS Option.

```
================================================================================
== E. QUALITATIVE STATISTICAL SUMMARY (QSS) ====================================
HR DEFAULT |  HR EXACT | RR DFLT  | RR EXACT  |  LACSLINK | EWS      | SUITELINK
       5830|      22554|         0|       1922|       1023|        28|         68
================================================================================
PS Form 3553, XXXXXXXX XXXX
```

For detailed information on the USPS Form 3553 (CASS Summary Report), refer to your *Working With Finalist Guide*.

CHAPTER 8

SuiteLink Option

This chapter describes the Finalist SuiteLink Option.

What is the SuiteLink Option?

The USPS SuiteLink database contains data on business addresses that were identified as default records that have associated secondary information during CASS processing. Finalist uses the USPS SuiteLink database to correct the secondary (suite) information in business addresses identified in the input file as high-rise default records. Records that have been processed through CASS Certified™ ZIP + 4® matching software and identified as defaults with secondary information are potential candidates for SuiteLink processing.

NOTE: USPS CASS regulations require SuiteLink processing for CASS certification and to generate the USPS Form 3553 (USPS CASS Summary Report).

Can I Use the SuiteLink Option?

The SuiteLink Option is available to all Finalist customers.

Why Should I Use the SuiteLink Option?

The SuiteLink Option provides you with the ability to improve deliverability for the business addresses in your mailing list. SuiteLink improves business address information by adding accurate secondary (suite) information to the business addresses in your mailing list. Adding this additional information makes USPS delivery sequencing available for addresses in your mailing list that previously were not eligible. Some of the benefits available to Finalist customers through SuiteLink processing are described below.

- Improves the business address information in your mailing list by appending secondary (suite) information to business addresses.
- Ensures USPS delivery sequencing for the business addresses in your mailing list.
- SuiteLink processing provides another method to ensure postal coding accuracy resulting in less undeliverable mail.

How Do I Install the SuiteLink Option?

The SuiteLink Option is installed as part of your standard Finalist installation. The distribution media sent to you in your release package contains all data needed to perform SuiteLink processing. After installing Finalist, you must activate the SuiteLink Option to perform SuiteLink processing.

How Do I Activate SuiteLink Option Processing?

To activate SuiteLink processing, you will need to define the fields listed below.

- **SuiteLink** — This field, in effect, turns on SuiteLink processing.
- **SuiteLink Filepath** — Defines the location of SuiteLink database.
- **SuiteLink Shutdown Indicator** — Indicates the action to take when encountering a SuiteLink processing error during the processing run.
- **Return SLK Input Secondary —** Indicates whether to return input secondary information when SuiteLink returns secondary information.
- **SuiteLink Small Memory Flag** — (Optional) Indicates the memory model to use for SuiteLink processing.

Methods for Activating SuiteLink Processing

To activate SuiteLink processing, use one of the following methods.

- **pbfn.cfg Configuration File** — If you prefer to configure your Finalist installation using global settings, you can define the SuiteLink fields in your pbfn.cfg configuration file. For more information on the pbfn.cfg file, refer to your *Working With Finalist Guide*.
- **PBFNSetupDef Structure** — If you prefer to configure your Finalist installation using the Finalist structures, you can define the SuiteLink fields in the PBFNSetupDef structure. For more information on the PBFNSetupDef structure, refer to your *Finalist Developer's Reference Guide*.
- **Workbench or Lookup Tool** — If you prefer to configure your Finalist installation using the Finalist Workbench or Lookup Tool GUI screens, you can define the SuiteLink fields in the **Product Tab** on the *PBFN Config Setting* dialog box. For more information on the Finalist Workbench and Lookup Tool, refer to your *Working With Finalist Guide*.
- **Compatibility Interface (CI)** — If you prefer to configure your Finalist installation using the CI, you can define the SuiteLink fields in the Finalist call area. For more information on the CI, refer to your *Finalist Developer's Reference Guide*.

Each method for activating SuiteLink processing is described in the following sections.

Using pbfn.cfg to Activate SuiteLink Processing

To activate SuiteLink processing using the pbfn.cfg file, complete the following fields in your pbfn.cfg file.

Table 1: pbfn.cfg File — SuiteLink Settings

pbfn.cfg Field	Description
SuiteLink Filepath	Specify the SuiteLink file name and path.
SuiteLink	Indicate whether to perform SuiteLink processing: • **OFF** — Do not perform SuiteLink processing. • **ON** — Perform SuiteLink processing. • blank — Defaults to **OFF**.
SuiteLink Small Memory Flag	Indicates the memory model for SuiteLink processing: • **P** — Pico. Stores no data in memory. No tables or indexes are loaded. • **U** — Ultra-small. Stores no data in memory. Partial indexes are loaded. • **S** — Small • **M** — Medium • **L** — Large • **H** — Huge. Stores all data in memory. blank — Defaults to **L**. Values from releases prior to the Finalist 8.0.0 release are accepted as: • If you specified ON, L is substituted. • If you specified OFF, U is substituted.

Table 1: pbfn.cfg File — SuiteLink Settings

pbfn.cfg Field	Description
SuiteLink Shutdown Indicator	Indicate the action to take when encountering a SuiteLink processing error during the processing run. Valid values are: • **W** — Issue warning and turn off SuiteLink processing. • **S** — Stop Finalist processing when encountering a SuiteLink error. • **I** — Ignore errors and continue SuiteLink processing. • blank — Defaults to **W**. **NOTE:** USPS regulations require SuiteLink processing for CASS certification and to generate a USPS Form 3553 (CASS Summary Report). Per the USPS regulations, any job that does not include SuiteLink processing, must not generate the USPS Form 3553 (CASS Summary Report). If you are processing in a CASS-certified mode (CASS=ON), Finalist sets the SuiteLink Shutdown Indicator to "S" in order to stop processing if the SuiteLink initialization fails and to prevent generation of an invalid USPS Form 3553 (CASS Summary Report).
Return SLK Input Secondary	Indicates how to return secondary information when SuiteLink secondary information is available. • **B** — Both. Return both SuiteLink and input secondary information. (This is the Default). • **S** — SuiteLink. Return SuiteLink secondary only. Do not return input secondary. • **I** — Input. Return input secondary only. Do not return SuiteLink secondary. • **N** — None. Do not return SuiteLink secondary or input secondary. • blank — Defaults to **B**. **NOTE:** If SuiteLink processing does not result in a match, Finalist ignores this option and returns the normal address output. Regardless of the option specified, the output ZIP + 4 is based on the match made using the SuiteLink secondary information. Input secondary information includes the "#" designator for purposes of this option.

Using the PBFNSetupDef Structure to Activate SuiteLink Processing

To activate SuiteLink processing using the PBFNSetupDef structure, complete the following fields in the PBFNSetupDef structure.

Table 2: PBFNSetupDef Structure — SuiteLink Settings

PBFNSetupDef Field	Description
cSuiteLinkFilePath	Specify the SuiteLink file name and path.
cAssignSuiteLink	Indicate whether to perform SuiteLink processing: • **OFF** — Do not perform SuiteLink processing. • **ON** — Perform SuiteLink processing. • blank — Defaults to **OFF**.

Table 2: PBFNSetupDef Structure — SuiteLink Settings

PBFNSetupDef Field	Description
cSuiteLinkSmallMem	Indicates the memory model for SuiteLink processing: • **P** — Pico. Stores no data in memory. No tables or indexes are loaded. • **U** — Ultra-small. Stores no data in memory. Partial indexes are loaded. • **S** — Small • **M** — Medium • **L** — Large • **H** — Huge. Stores all data in memory. blank — Defaults to **L**. Values from releases prior to the Finalist 8.0.0 release are accepted as: • If you specified ON, L is substituted. • If you specified OFF, U is substituted.
cSuiteLinkShutdown	Indicate the action to take when encountering a SuiteLink processing error during the processing run. Valid values are: • **W** — Issue warning and turn off SuiteLink processing. • **S** — Stop Finalist processing when encountering a SuiteLink error. • **I** — Ignore errors and continue SuiteLink processing. • blank — Defaults to **W**. **NOTE:** USPS regulations require SuiteLink processing for CASS certification and to generate a USPS Form 3553 (CASS Summary Report). Per the USPS regulations, any job that does not include SuiteLink processing, must not generate the USPS Form 3553 (CASS Summary Report). If you are processing in a CASS-certified mode (CASS=ON), Finalist sets the SuiteLink Shutdown Indicator to "S" in order to stop processing if the SuiteLink initialization fails and to prevent generation of an invalid USPS Form 3553 (CASS Summary Report).
cRetSLKinputSecdry	Indicates how to return secondary information when SuiteLink secondary information is available. • **B** — Both. Return both SuiteLink and input secondary information. (This is the Default). • **S** — SuiteLink. Return SuiteLink secondary only. Do not return input secondary. • **I** — Input. Return input secondary only. Do not return SuiteLink secondary. • **N** — None. Do not return SuiteLink secondary or input secondary. • blank — Defaults to **B**. **NOTE:** If SuiteLink processing does not result in a match, Finalist ignores this option and returns the normal address output. Regardless of the option specified, the output ZIP + 4 is based on the match made using the SuiteLink secondary information. Input secondary information includes the "#" designator for purposes of this option.

Using the Finalist Workbench to Activate SuiteLink Processing

To activate SuiteLink processing using the Finalist Workbench:

1. Launch the Finalist Workbench.
2. From the **Tools** menu, select **PBFN Setup**.
3. Select the **Product** tab on the *PBFN Config Setting* dialog box.

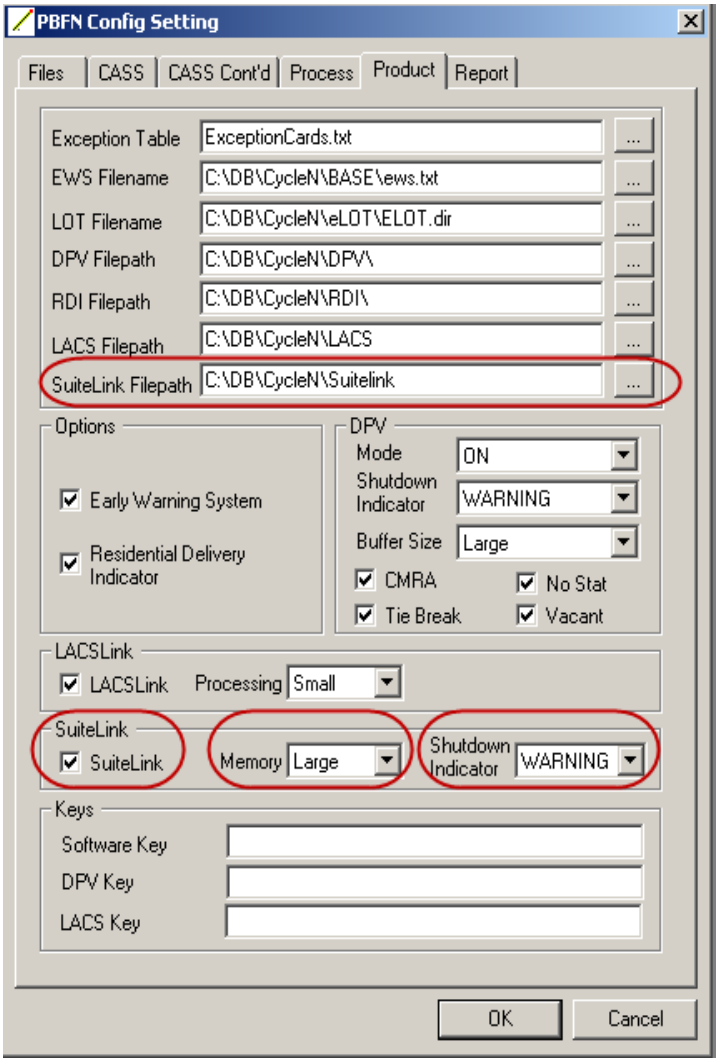

Figure 1: PBFN Config Setting Dialog Box — Product Tab

4. Complete the following fields on the **Product** tab.

Table 3: PBFN Config Setting Dialog Box — Product Tab

Field	Description
SuiteLink Filepath	Specify the SuiteLink file name and path or click on the **Browse** button (located to the right of the field) to open a dialog box and point to the folder (path) that contains the databases.
SuiteLink	Check the SuiteLink box to perform SuiteLink processing.
Memory	Indicates the memory model for SuiteLink processing: • **Pico** — Pico. Stores no data in memory. No tables or indexes are loaded. • **Ultra** — Ultra-small. Stores no data in memory. Partial indexes are loaded. • **Small** — Small • **Medium** — Medium • **Large** — Large • **Huge** — Huge. Stores all data in memory. blank — Defaults to **Large**. Values from releases prior to the Finalist 8.0.0 release are accepted as: • If you specified ON, Large is substituted. • If you specified OFF, Ultra is substituted.
Shutdown Indicator	Select the action to take when encountering a SuiteLink processing error during the processing run. Valid values are: • **WARNING** — Issue warning and turn off SuiteLink processing. • **SHUTDOWN** — Stop Finalist processing when encountering a SuiteLink error. • **IGNORE** — Ignore errors and continue SuiteLink processing. • blank — Defaults to **WARNING**. **NOTE:** USPS regulations require SuiteLink processing for CASS certification and to generate a USPS Form 3553 (CASS Summary Report). Per the USPS regulations, any job that does not include SuiteLink processing, must not generate the USPS Form 3553 (CASS Summary Report). If you are processing in a CASS-certified mode (CASS=ON), Finalist sets the SuiteLink Shutdown Indicator to "S" in order to stop processing if the SuiteLink initialization fails and to prevent generation of an invalid USPS Form 3553 (CASS Summary Report).

5. Select the **Process** tab on the *PBFN Config Setting* dialog box.

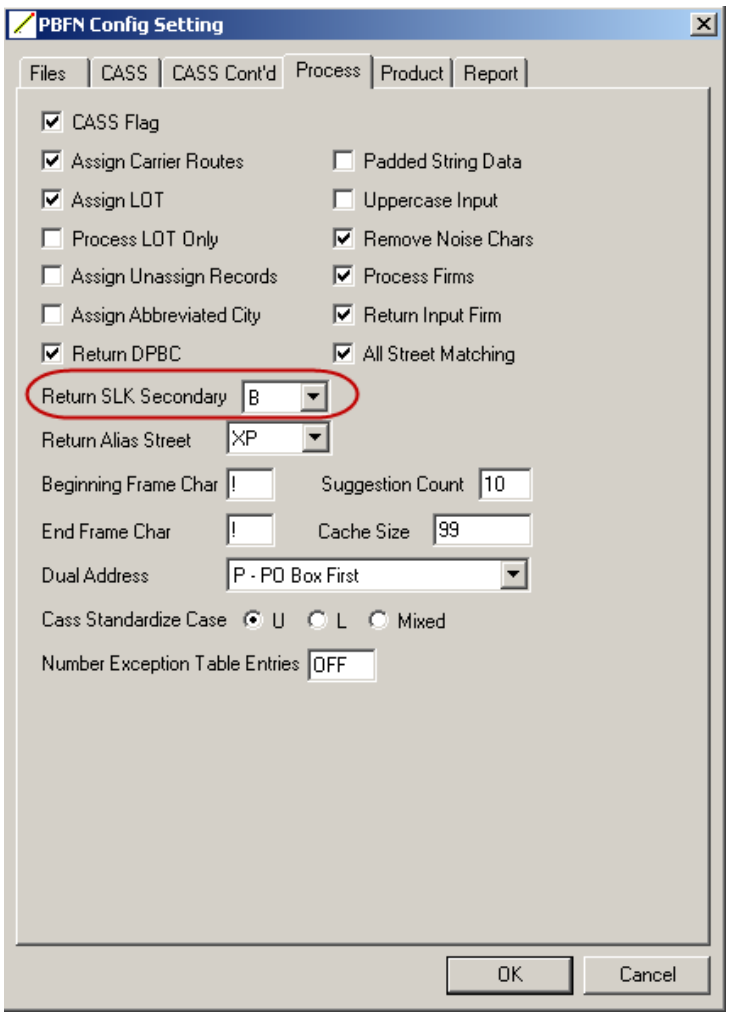

Figure 2: PBFN Config Setting Dialog Box — Process Tab

6. Complete the following fields on the **Process** tab.

Table 4: PBFN Config Setting Dialog Box — Process Tab

Field	Description
Return SLK Secondary	Indicates how to return secondary information when SuiteLink secondary information is available. • **B** — Both. Return both SuiteLink and input secondary information. (This is the Default). • **S** — SuiteLink. Return SuiteLink secondary only. Do not return input secondary. • **I** — Input. Return input secondary only. Do not return SuiteLink secondary. • **N** — None. Do not return SuiteLink secondary or input secondary. • blank — Defaults to **B**. **NOTE:** If SuiteLink processing does not result in a match, Finalist ignores this option and returns the normal address output. Regardless of the option specified, the output ZIP + 4 is based on the match made using the SuiteLink secondary information. Input secondary information includes the "#" designator for purposes of this option.

7. Click **OK** to save your settings.

Using the Compatibility Interface to Activate SuiteLink Processing

To activate SuiteLink processing using the CI, define the appropriate field for your platform in the Finalist call area.

Table 5: Compatibility Interface (CI) - SuiteLink Settings

COBOL Field Name/ C Field Name	Field Description
FINAL-SLK-OPT caSLK	Determines whether Finalist performs SuiteLink processing. If this field contains a "Y", Finalist performs SuiteLink processing.

The SuiteLink memory options cannot be specified using the CI. The CI uses the default SuiteLink memory model (Large memory model).

How Does SuiteLink Option Processing Work?

Finalist performs SuiteLink processing as described below.

1. Finalist calls SuiteLink when the following conditions are met:
   - The Finalist configuration file (pbfn.cfg) indicates SuiteLink = ON and all other required SuiteLink parameters are defined with valid values.
   - Finalist successfully coded the address and the following information exists in the address record:
     - Firm name
     - Valid ZIP Code
     - Valid ZIP + 4 Code
     - Primary number exists
   - A match has been made to a high-rise or street default record.
   - The Finalist database is current.
   - The SuiteLink database is current.
2. If SuiteLink returns secondary data, Finalist performs another match attempt using the corrected data.
3. Finalist checks the Return SLK Input Secondary configuration setting to determine how to return secondary information when SuiteLink secondary information is available.
4. Finalist prints statistics at end of job.

Structures Containing SuiteLink Option Information

The structures listed below include data to facilitate SuiteLink processing. For detailed information on these structures, refer to your *Finalist Developer's Reference Guide*.

- PBFN3553Def
- PBFNInfoDef
- PBFNParsedAdrAltDef
- PBFNParsedAdrDef (includes SuiteLink return code)
- PBFNProcessDataAltDef
- PBFNProcessDataDef (includes SuiteLink return code)
- PBFNRtnSuiteLinkStatsDef (returns SuiteLink processing statistics)
- PBFNSetupDef (includes three SuiteLink setup parameters)

SuiteLink Option Output

This following sections describe the output generated during SuiteLink processing.

SuiteLink Error Messages

If SuiteLink generates an error, Finalist outputs an error message to the Finalist log file.

SuiteLink Return Codes

If SuiteLink processing changes an address, the cSteLnkRtnCode field in the PBFNProcessDataDef structure (or PBFNProcessDataAltDef for languages like COBOL that need to use character arrays in place of pointers for data fields) contains one of the return codes listed below.

Table 6: SuiteLink Option Error Messages

Return Code	Description
A	SuiteLink processing successful. Record matched through SuiteLink processing.
00	SuiteLink processing failed. No matching record found during SuiteLink processing.

SuiteLink Information on Finalist Batch Report

The Finalist Batch Report displays the SuiteLink settings from the pbfn.cfg configuration file or the PBFNSetupDef structure and the SuiteLink processing counts. For detailed information on the Finalist Batch Report, refer to your *Working With Finalist Guide*.

SuiteLink Information on USPS Form 3553 (CASS Summary Report)

The Qualitative Statistical Summary (QSS) section of the USPS Form 3553 (CASS Summary Report) displays as shown below if you are processing with the SuiteLink Option.

```
================================================================================
== E. QUALITATIVE STATISTICAL SUMMARY (QSS) ====================================
HR DEFAULT |  HR EXACT | RR DFLT  | RR EXACT  |  LACSLINK | EWS      | SUITELINK
       5830|      22554|         0|       1922|       1023|        28|         68
================================================================================
PS Form 3553, XXXXXXXX XXXX
```

For detailed information on the USPS Form 3553 (CASS Summary Report), refer to your *Working With Finalist Guide*.

CHAPTER 9

Exceptions Table Option

This chapter describes the Finalist Exceptions Table Option. You can use the Exceptions Table Option to alter address input fields to enhance the coding ability of Finalist.

What is the Exceptions Table Option?

The exceptions table alters address input fields to enhance Finalist's coding ability. According to rules that you establish, the exceptions table modifies addresses that may be rejected so that the addresses match addresses found in the Finalist Data File. One potential use would be to look for commonly misspelled streets or city names on input and replace the misspelled name with the corrected name. These changes do not affect your input data. Be careful when making exceptions table entries. An exceptions table entry affects all records that meet the selection criteria. Again, the exceptions table does not change the input data, but rather the data that Finalist processes.

Can I Use the Exceptions Table Option?

The Exceptions Table Option is available to all Finalist customers.

Why Should I Use the Exceptions Table Option?

Some of the benefits available to Finalist customers through Exceptions Table processing are:

- Provides a method to change input address fields to increase coding
- Allows you to set rules for address modification using the exceptions table to ensure address matching
- You can use the exceptions table to:
  - Add missing suffixes to addresses
  - Combine two words
  - Correct misspelled street names
  - Ignore extraneous address information
  - Make address changes within the items listed next:
    - A specific ZIP Code
    - An SCF (first three digits of ZIP Code) area
    - A region (first digit of ZIP Code)
    - All input addresses

How Do I Install the Exceptions Table Option?

The Exceptions Table Option is installed as part of your standard Finalist installation. The distribution media sent to you in your release package contains all data needed to perform Exceptions Table processing. After installing Finalist, you must activate the Exceptions Table Option to perform Exceptions Table Option processing.

How Do I Activate Exceptions Table Processing?

To activate Exceptions Table processing, you will need to define the fields listed below.

- **Exception Table Filename** — Define the Exceptions Table file name and path.
- **Number Exception Table Entries** — Specify the maximum number of exceptions table entries.

Methods for Activating Exceptions Table Processing

To activate Exceptions Table processing, use one of the following methods.

- **pbfn.cfg Configuration File** — If you prefer to configure your Finalist installation using global settings, you can define the Exceptions Table fields in your pbfn.cfg configuration file. For more information on the pbfn.cfg file, refer to your *Working With Finalist Guide*.
- **PBFNSetupDef Structure** — If you prefer to configure your Finalist installation using the Finalist structures, you can define the Exceptions Table fields in the PBFNSetupDef structure. For more information on the PBFNSetupDef structure, refer to your *Finalist Developer's Reference Guide*.
- **Workbench or Lookup Tool** — If you prefer to configure your Finalist installation using the Finalist Workbench or Lookup Tool GUI screens, you can define the Exceptions Table fields in the **Product Tab** on the *PBFN Config Setting* dialog box. For more information on the Finalist Workbench and Lookup Tool, refer to your *Working With Finalist Guide*.
- **Compatibility Interface (CI)** — If you prefer to configure your Finalist installation using the CI, you can define the Exceptions Table fields in the Finalist call area. For more information on the CI, refer to your *Finalist Developer's Reference Guide*.

Each method for activating Exceptions Table processing is described in the following sections.

Using pbfn.cfg to Activate Exceptions Table Processing

To activate Exceptions Table processing using the pbfn.cfg file, complete the following fields in your pbfn.cfg file.

Table 1: pbfn.cfg File — Exceptions Table Settings

pbfn.cfg Field	Description
Exception Table Filename	Specify the Exceptions Table file name and path.
Number Exception Table Entries	Specify the maximum number of exceptions table entries. The default value is **1000**.

Using the PBFNSetupDef to Activate Exceptions Table Processing

To activate Exceptions Table processing using the PBFNSetupDef structure, complete the following fields in the PBFNSetupDef structure.

Table 2: PBFNSetupDef Structure — Exceptions Table Settings

PBFNSetupDef Field	Description
cExceptionTblFileName	Specify the Exceptions Table file name and path.
cMaxExcpTblEntries	Specify the maximum number of exceptions table entries. The default value is **1000**.

Using the Workbench to Activate Exceptions Table Processing

To activate Exceptions Table processing using the Finalist Workbench:

1. Launch the Finalist Workbench.
2. From the **Tools** menu, select **PBFN Setup**.

3. Select the **Product** tab on the *PBFN Config Setting* dialog box.

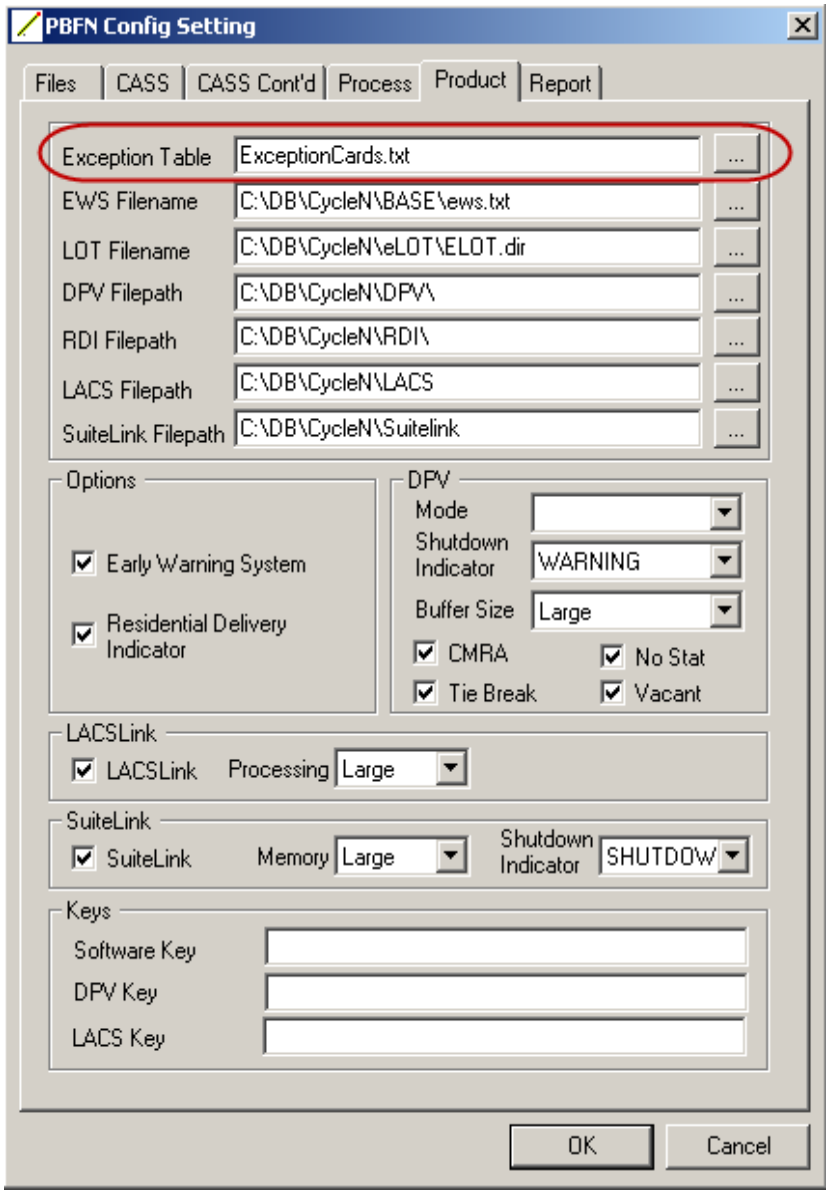

Figure 1: PBFN Config Setting Dialog Box — Product Tab

4. Complete the following field on the **Product** tab.

Table 3: PBFN Config Setting Dialog Box — Product Tab

Field	Description
Exception Table File	Specify the Exceptions Table File file name and path or click on the **Browse** button (located to the right of the field) to open a dialog box and select the path and file name from a list of existing files.

5. Select the **Process** tab on the *PBFN Config Setting* dialog box.

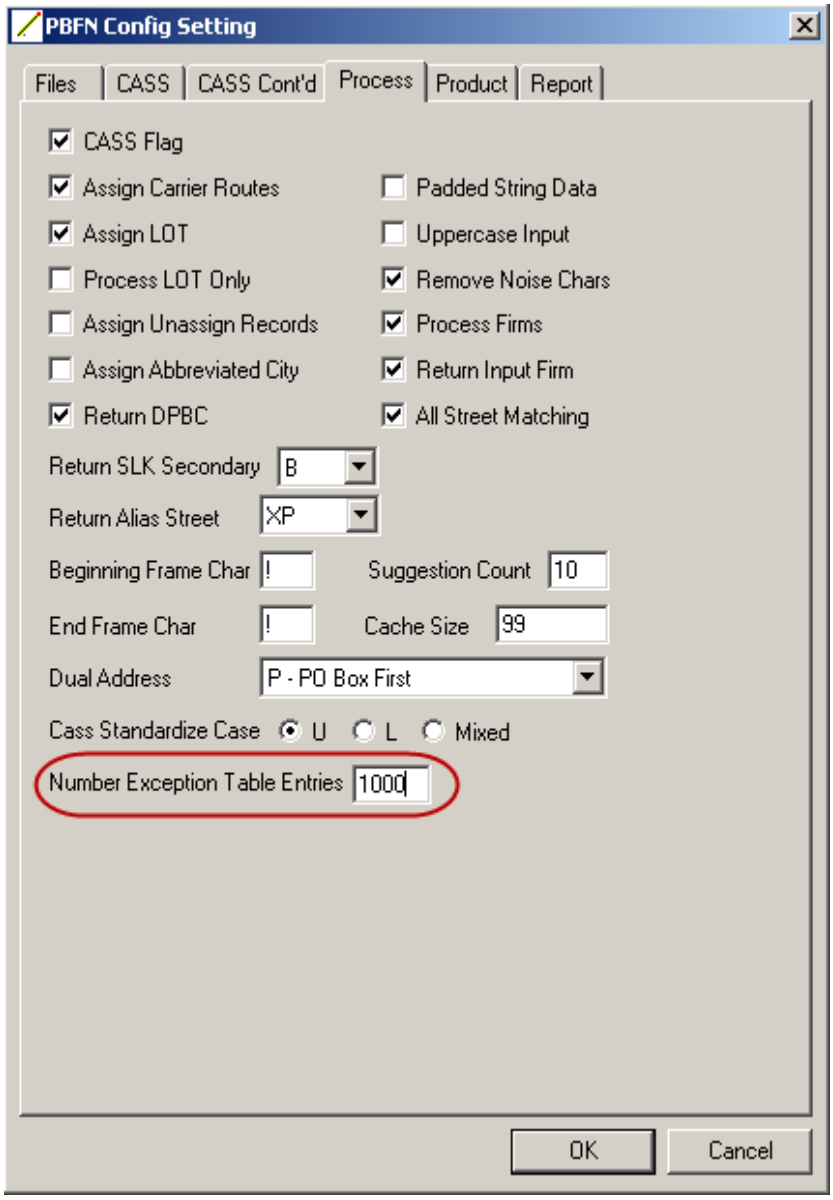

Figure 2: PBFN Config Setting Dialog Box — Process Tab

6. Complete the following field on the **Process** tab.

Table 4: PBFN Config Setting Dialog Box — Process Tab

Field	Description
Number of Exceptions Table Entries	Specify the maximum number of exceptions table entries. The default value is **1000**.

7. Click **OK** to save your settings.

Using the Compatibility Interface to Activate Exceptions Processing

To activate Exceptions Table processing using the CI, define the appropriate field for your platform in the Finalist call area.

Table 5: Compatibility Interface (CI) - Exceptions Table Settings

COBOL Field Name/ C Field Name	Field Description
CAEXCPSW cExcpSw	Applies to CICS users only. CAEXCPSW indicates exceptions table processing is used. The field is populated only on the initialization call.

How Does Exceptions Table Processing Work?

Finalist performs Exceptions Table as described below.

1. You create an exceptions table in an ASCII text format using any text editor.

   NOTE: ALL exceptions should be in alphabetical order.

2. During processing of your input file, Finalist uses your exceptions table file to apply changes to addresses in your input file that could be rejected to ensure a match to the Finalist Data File.

3. The exceptions table does not change the input data, but rather the data that Finalist processes.

4. The Address Detail Report indicates whether an exceptions table entry was used to process an address.

5. The number of entries in your exceptions table displays on the Finalist Batch Report. If exceptions table processing is not activated, the "No. Exception Table Entries" field displays OFF. The Exceptions Table filename also displays on the Finalist Batch Report.

Exceptions Table Input

To use an exceptions table, you can use any text editor to create a file in an ASCII text format. Record lines must be 80 characters in length. Table 6 describes the fields available for an exceptions table entry. All field positions need to be specified exactly as shown.

Table 6: Exceptions Table Entry Fields and Positions

Field Name	Position	Description
L#	1-2	This required field designates the address line that the entry affects. Possible values for L# are described next. L1 = First address line L2 = Second address line L3 = City/state line
ARGU	3 - 17	This required field contains a one-word argument (keyword) or the first word of a two-word argument for which you want Finalist to search.
SUBST A	18 - 32	This required field contains the word that replaces, or is used with, ARGU. For an IG-type entry, leave this field blank. Blanks are only valid with an IG-type entry. For a ++-type entry, this field becomes the second word of a two-word argument.
ATTRIBUTE	33 - 34	This required field contains the two-digit code to supply more information about the substitute word or the change being made on the argument word.
ZIP Region (first digit of ZIP Code) SCF (first 3 digits of ZIP Code) ZIP Code	 35 35 - 37 35 - 39	This optional field specifies the region, SCF, or ZIP Code for which the exception logic works. If this field is blank, exception logic works for all input data.
SUBST B	40 - 54	This optional field contains the replacement for ARGU and SUBST A in a ++-type exceptions table entry. This optional field is also used in the R+ operation, which concatenates SUBST B with SUBST A as a replacement for ARGU.

Table 7 provides a list of available attributes for each address line.

Table 7: Attributes for Address Lines

Address Line(s)	Attribute	Description
L1 and L2	DR	Replace argument word with substitute word. The substitute word is a directional.
	I2	Insert the substitute word as a unit designator before the target word.
	ID	Insert the substitute word as a directional before the argument word.
	IN	Insert the substitute word after the argument and make it part of the street name.
	IP	Insert the substitute word as a directional after the argument word.
	IS	Insert the substitute word as a suffix after the argument word.
	IU	Insert the substitute word as a unit designator after the target word.
	SF	Replace argument word with substitute word. The substitute word is a suffix.
	UD	Replace argument word with the substitute word. The substitute word is a unit designator.
	RR	Replace argument word with the substitute word. The substitute word is a Rural Route (RR) word.
L3	IC	Insert the substitute word as a city before the argument state abbreviation.
	ST	Insert the substitute word as the state abbreviation after the argument city word.
L1, L2, and L3	++	Combine the argument and substitute words to make one word.
	CT	Attach (concatenate) the substitute word to the following word in the input record.
	IG	Ignore the argument word.
	R+	Replace argument word with the concatenation of substitute word A and substitute word B.
	RP	Replace argument word with substitute word.

Entry Guidelines

Follow these guidelines when making exceptions table entries.

- There is a default of a 1,000-entry maximum for the exceptions table. Sort the entries sequentially by argument, in ascending order. Alphabetic arguments should precede numeric arguments.

EXAMPLE

```
    1...............18.............33.....
 1) L3ADDISON        IL             ST60101
 2) L1AMERICAN       LEGION         IN02131
 3) L1BEAVER         TER            IS01701
    .
    .
15) L1U              U              CT53913
16) L2ZEPHYR         N              ID79963
17) L135TH           W              IP99821
```

- If you use SCFs, regions, or ZIP Codes to limit alterations for an exceptions statement, you must pass a ZIP Code field as input to Finalist.
- For any function that performs a lookup on city/state, L3 alterations could be useful.

Sample Entries

Sample applications using exceptions table logic are shown next.

Table 8: Sample Applications of Exceptions Table Logic With ++ Attribute

Attribute	Description
++	Combine the argument and substitute words to make one word.

Table 8: Sample Applications of Exceptions Table Logic With ++ Attribute

Attribute	Description
Sample Application 1	BAYVIEW exists in ZIP Code 60532. Your input file lists BAY VIEW. With the following exceptions statement, Finalist looks for records with the argument word BAY and the substitute word VIEW on the same line. The single word BAYVIEW (coded after the ZIP Code field) replaces both words.
Input	`13 BAY VIEW 60532`
Exception	`1BAY VIEW ++60532BAYVIEW` This attribute combines words more specifically than the CT attribute. Instead of combining the argument word with whatever follows, ++ looks for specific word pairs.
Sample Application 2	STATE ROUTE 59 exists in ZIP Code 60185. It is commonly known as, and is listed in your input file as, ILLINOIS 59. With the following exceptions statement, the Finalist system looks for records with the argument word ILLINOIS and the second argument 59 on the same line. The single string STATEROUTE59 (coded after the ZIP Code field) replaces both words.
Input	`1N601 ILLINOIS 59`
Exception	L1ILLINOIS 59 ++60185STATE ROUTE 59 This example shows when you should use the ++ attribute instead of the RP (replace) attribute. Your purpose in using the ++ attribute is to replace the word ILLINOIS only when it is paired with the trailing word 59. Here, the ++ attribute replacement for ILLINOIS 59 would be STATE ROUTE 59. If you use the RP attribute to replace Illinois with State Route, Illinois will be changed to State Route whenever Illinois is encountered (i.e., Illinois Ave. becomes State Route Ave).

Table 9: Sample Applications of Exceptions Table Logic With CT Attribute

Attribute	Description
CT	Concatenate the substitute word (which is also the argument word in most cases) with the word that follows it on input.
Sample Application 1	BROADVIEW exists as a street in ZIP Code 60148. Your file lists this street as BROAD VIEW. You could use the following exceptions statement to correct the problem:
Input	`10 BROAD VIEW 60148`
Exception	`L1BROAD BROAD CT60148` The system can now locate the street on the Data File and code the record. Finalist doesn't alter the input record, but will return BROADVIEW in both the returned street name field and the label return area. **NOTE:** All occurrences of BROAD will join with street name words that follow BROAD in the specified ZIP Code.
Sample Application 2	You can also use CT to make multiple changes. In Example 1 above, if your file spells BROADVIEW incorrectly, you can change the spelling and combine the words.
Input	`10 BOARD VIEW 60148`
Exception	`L1BOARD BROAD CT60148` The system replaces BOARD with BROAD and concatenates it to VIEW. Finalist returns BROADVIEW in the returned street name field and the label return area.

Table 10: Sample Application of Exceptions Table Logic With DR Attribute

Attribute	Description
DR	Replace argument with substitute and identify the substitute word as a directional.
Sample Application	Some of the records in your file incorrectly use NROTH to represent NORTH. The following exceptions statement corrects the problem:
Input	`12 NROTH MAIN ST 60148`
Exception	`L1NROTH N DR` The system replaced NROTH with N. Finalist will then be able to code to N MAIN ST. Finalist will then return N in the pre-directional field of the Finalist return area and the label return area. If Finalist replaces a post-directional, it returns in the post-directional field and the label return area.

Table 11: Sample Application of Exceptions Table Logic With I2 Attribute

Attribute	Description
I2	Insert the substitute word as a unit designator before the target word.
Sample Application	The address 730 GRAND CNCRS AB is missing APT before the unit number (AB). Create the following exceptions table entry to insert APT before AB in ZIP code 10451:
Input	`730 GRAND CNCRS AB 10451`
Exception	`L1AB APT I210451` The exception logic inserts APT to effectively process the address as 730 GRAND CNCRS APT AB.

Table 12: Sample Application of Exceptions Table Logic With ID Attribute

Attribute	Description
ID	Insert a directional before street name (pre-directional).
Sample Application	The address 200 BERKSHIRE needs a directional to code. This exceptions statement adds the proper directional:
Input	`200 BERKSHIRE AVE 60148`
Exception	`L1BERKSHIRE E ID60148` The system can now code the record to E BERKSHIRE. The input address remains BERKSHIRE. Finalist returns E in the return area's pre-directional field and the label return area.

Table 13: Sample Application of Exceptions Table Logic With IG Attribute

Attribute	Description
IG	Ignore the argument (here you use no substitute word).
Sample Application	COUNTRY CORNERS RD exists in ZIP Code 01002 and the USPS lists it this way. However, your file lists this street as HWY COUNTRY CORNERS RD. The following exceptions statement makes the system recognize HWY COUNTRY CORNERS RD as COUNTRY CORNERS RD:
Input	`31 HWY COUNTRY CORNERS RD 01002`
Exception	`L1HWY IG01002` The system now ignores HWY during coding. Your input record remains intact. The street name that Finalist returns in the return area is COUNTRY CORNERS RD. Finalist also correctly standardizes the label return area.

Table 14: Sample Application of Exceptions Table Logic With IN Attribute

Attribute	Description
IN	Insert the substitute word after the argument and use it as part of the street name.
Sample Application	AMERICAN LEGION HWY exists in ZIP Code 02131. Your input file has it listed as AMERICAN HWY. The following exceptions statement corrects the problem:
Input	`865 AMERICAN HWY 02131`
Exception	`L1AMERICAN LEGION IN02131` The record now codes as AMERICAN LEGION HWY and Finalist returns that name in the return area street field and the label return area.

Table 15: Sample Application of Exceptions Table Logic With IP Attribute

Attribute	Description
IP	Insert the substitute word as a directional after the street name (post-directional).
Sample Application	The address 1 DAVIS RD needs a post-directional to code. Use the following exceptions statement to add the necessary post-directional:
Input	`1 DAVIS RD 06371`
Exception	`L1DAVIS E IP06371` The exception logic lets the record code to DAVIS RD E. The input remains unchanged. Finalist returns E in the post-directional field and in the return area label area.

Table 16: Sample Application of Exceptions Table Logic With IS Attribute

Attribute	Description
IS	Insert a suffix after the street name.
Sample Application	BEAVER TER CIR exists in ZIP Code 01701. Your input record lists BEAVER CIR and will not code accurately. The following exceptions statement corrects the problem:
Input	`100 BEAVER CIR 01701`
Exception	`L1BEAVER TER IS01701`
	The record now codes correctly without changing the input record. If you are using parsed fields, this may be a concern. However, the suffix1 field in the Finalist return area contains TER and the suffix2 field contains CIR. The label return area contains 100 BEAVER TERRACE CIR.

Table 17: Sample Application of Exceptions Table Logic With IU Attribute

Attribute	Description
IU	Insert the substitute word as a unit designator after the target word.
Sample Application	The address 730 GRAND CNCRS AB is missing APT after the street word (CNCRS) and before the unit (AB). Create the following exceptions table entry to insert APT following CNCRS in ZIP code 10451:
Input	`730 GRAND CNCRS AB 10451`
Exception	`L1CNCRS APT IU10451`
	The exception logic inserts APT to effectively process the address as 730 GRAND CNCRS APT AB.

Table 18: Sample Applications of Exceptions Table Logic With RP Attribute

Attribute	Description
RP	Replace the argument word with a substitute.
Sample Application 1	LEXINGTON AVE exists in New York City but is commonly known as LEX AVE. Your file contains this reference, but is not a sufficient match for LEXINGTON. To code this record, use the following exception statement:
Input	`142 LEX AVE 10016`
Exception	`L1LEX LEXINGTON RP10016`
	Finalist will leave LEX in your input record. It will return LEXINGTON in the return area street field and the label area. Reason code 5 will have a value of 2.
Sample Application 2	If your file misspells a state abbreviation (for example, II for IL), you can use the following exceptions statement to correct the problem:
Input	`500 WEST RD LOMBARD II`
Exception	`L3II IL RP`
	Without altering the input data, the exception logic uses IL instead of II. The FINAL subroutine codes the record. Since II is not a valid city or state, no ZIP Code, SCF, or region is necessary.
Sample Application 3	If your file contains a non-standard city abbreviation (for example, CHG for CHICAGO), the following exceptions statement corrects the problem:
Input	`600 W MADISON ST CHG IL 60602`
Exception	`L3CHG CHICAGO RP606`
	The city of Chicago uses the SCF 606 exclusively. If you do not pass a ZIP Code to Finalist, you do not need an SCF in the exceptions statement.
Sample Application 4	If a directional and a street name are combined (i.e., NORTHVIEW), you need two exception entries to correct the problem.
Input	`3109 NORTHVIEW ROCKFORD IL 61107`
Exception	`L1NORTHVIEW VIEW RP61107` `L1NORTHVIEW N ID61107`
	The first entry replaces the double word with the correct primary street name. The second entry inserts the correct directional.

Table 19: Sample Application of Exceptions Table Logic With R+ Attribute

Attribute	Description
R+	Replace the argument word with a substitute.
Sample Application	ADAM CLAYTON POWELL JR BLVD exists in New York City but does not fit into the input address file space allowed for the street name. Create the following exceptions table entry to expand ACP to ADAM CLAYTON POWEL JR in ZIP code 10026:
Input	`1800 ACP BLVD 10026`
Exception	`L1ACP ADAM CLAYTON POR+10026WELL JR`
	Finalist leaves ACP in your input record and returns ADAM CLAYTON POWELL JR in the return area street field and the label area.

Table 20: Sample Application of Exceptions Table Logic With RR Attribute

Attribute	Description
RR	Replace the argument word with a substitute. The substitute word is a Rural Route.
Sample Application	RURAL ROUTE is misspelled as RVRALRTE in ZIP 77622. Create the following exceptions table entry to convert RVRALRTE to RR in ZIP code 77622:
Input	`RVRALRTE 2 BOX 703 77622`
Exception	`L1RVRALRTE RR RR77622`
	Finalist leaves RVRALRTE in your input record and returns RR in the return area street field and the label area. In this sample, RR 2 BOX 703 will LACS^{Link} convert to 12396 E HAMPSHIRE RD.

Table 21: Sample Application of Exceptions Table Logic With SF Attribute

Attribute	Description
SF	Replace the argument word with the substitute words which the system will identify as a suffix.
Sample Application	DRIVE is a street suffix and DR is its recognized abbreviation. DV, a non-recognized abbreviation, appears in your file in both address lines. The following exceptions statements help Finalist recognize DV as a suffix when it appears in either address line. This exceptions statement is especially important when the suffix is crucial to coding.
Input	`2100 BELLEAU WOODS DV 60187`

Exception	`L1DV DR SF`
	`L2DV DR SF`
	
	The system will now recognize DV as the suffix DR and Finalist can code these records. Please note that because both DR and CT are valid for the above range, the exceptions logic could affect coding if you altered the improper suffix. Using the above exceptions logic, Finalist will leave DV in your input record, and return DR in the Finalist return area suffix field. The label area will also contain this correctly standardized suffix.

City/State Only Lookups

The following two exceptions table attributes are useful for city/state only lookup. Determining whether to use region, SCF, or ZIP Code limitations in the exceptions statement depends on whether you'll pass the input ZIP Code field with the city/state field. If you are not including a ZIP Code in the input, do not use a ZIP Code in the exceptions statement.

NOTE: In these examples, Finalist alters any word(s) located in the specified ZIP Code that contains the argument word.

Table 22: Sample Application of Exceptions Table Logic With IC Attribute

Attribute	Description
IC	Insert the substitute word as the city before the argument state word.
Sample Application	Use this type of alteration only if you're passing ZIP Codes in the ZIP Code field and passing the city/state field (since you only want to insert a city for certain ZIP Codes within the state). For example, a few records at ZIP Code 60104 have only IL in the city/state field. If you only use a city/state lookup for your file, this record would fail because the city is missing. Use the following exception statement to insert a city in your record:
Exception	`L3IL BELLWOOD IC60104` This statement inserts the city BELLWOOD in all your IL records for the ZIP Code 60104. Function 7 is now able to code the address. Functions 4, 5, and 6 do not need a city name to code.

Table 23: Sample Application of Exceptions Table Logic With ST Attribute

Attribute	Description
ST	Insert this as the state.
Sample Application	If you perform a city/state-only lookup, some records might fail because the state name is missing. Use an exceptions table entry to insert the state for these records. Since identical city names exist in different states, it is safest to restrict this alteration to a specific SCF or ZIP Code.
Input	`22 E LAKE ST ADDISON 60101`
Exception	`L3ADDISON IL ST60101` The system uses IL for the state and your input city of ADDISON. The record codes.

Table 24: Sample Application of Exceptions Table Logic With UD Attribute

Attribute	Description
UD	Replace the argument word with a substitute and flag as a unit designator.
Sample Application	You use the word LT as an abbreviation for the unit designator LOT. Finalist will leave the word spelled as LT but flag the word as a unit designator.
Exception	`L1LT LT UD`

Structures Containing Exceptions Table Information

The structures listed below include data to facilitate Exceptions Table processing. For detailed information on these structures, refer to your *Finalist Developer's Reference Guide*.

- PBFNAddressInfoDef
- PBFNIMSSetupDef
- PBFNSetupDef

Exceptions Table Output

This following sections describe the output generated during Exceptions Table processing.

Exceptions Table Information on Finalist Batch Report

The Finalist Batch Report displays the Exceptions Table settings from the pbfn.cfg configuration file or the PBFNSetupDef structure and the number of entries in your Exceptions Table. If Exceptions Table processing is not activated, the "No. Exceptions Table Entries" field displays OFF. For detailed information on the Finalist Batch Report, refer to your *Working With Finalist Guide*.

Exceptions Table Information on the Address Detail Report

The Address Detail Report indicates whether an Exceptions Table entry was used to process an address. For detailed information on the Address Detail Report, refer to your *Working With Finalist Guide*.

CHAPTER 10

ADDRSCAN Option

This chapter describes the ADDRSCAN API including ADDRSCAN benefits and features, processing results, function structure, and how to call ADDRSCAN from a user-written driver.

What is the ADDRSCAN Option?

You can use the ADDRSCAN function to:

- Identify individual components within a multiple-line address record
- Format and standardize the address information using United States Postal Service (USPS) logic
- Return the processed address lines in the order you specify
- Identify firm, street, and city lines, so you can utilize the best combination of data from unstructured lists or multi-use lists to reformat and restructure your address files
- Pre-process addresses that contain multiple lines or "floating" lines

---

NOTE: A floating line is an address line or city/state/ZIP Code line whose position appears in different parts of an address record. This line moves, or "floats", to any of the fixed number of fields allocated to a name/address record. The following example illustrates the concept of floating address lines.

EXAMPLE

In the first example address, the PO box is located on the first line. In the second example address, the PO box is located on the second line. The position of the PO box data moves or "floats" in the address record.

```
PO Box 99813
654 S Lawndale
Roseville, MI 48066

654 S Lawrence Ave
PO Box 9981B
Roseville, MI 48066
```

ADDRSCAN lets you concatenate multiple address lines and return two address lines, a city/state line, a firm line, and a 10-byte ZIP + 4 field from the floating address lines. By returning fixed lines for processing, you eliminate the need to search for address, city/state, and ZIP Code components.

NOTE: As stated above, ADDRSCAN attempts to pre-process addresses that contain multiple or floating lines. ADDRSCAN cannot extract address information in the middle of other information on the same line. Also, ADDRSCAN may not be able to process address information spread across non-contiguous lines.

The following is an example of address data that cannot be processed clearly.

```
2200 WESTERN CT
ED'S PLACE IS APT 100
ACCOUNT IN QUESTION 12345
LISLE IL 60532
3618
```

Why Should I Use the ADDRSCAN Option?

You can use ADDRSCAN to:

- Consistently identify the appropriate address lines to pass to a postal coding API such as Finalist. This ensures the best chance of coding
- Restructure your address lines
- Identify the ZIP Code and the ZIP + 4 code if present on input
- Identify and return Puerto Rico URB data
- Specify the format of the returned lines
- Return the original address lines in a standardized format and identify each line
- Accept up to six address lines as input and return four address lines and a 10-byte ZIP Code field
- Use your user-written driver to call ADDRSCAN to identify and/or reorganize address lines before passing the data to a postal coding API such as Finalist
- Relate the returned line address to both your original input address and the standardized address
- Set up exceptions for ADDRSCAN processing on the mainframe platform using the ADDRTBLS module (not available on other platforms)

The list below provides specific examples of how the ADDRSCAN function benefits Finalist customers.

- Your input address record contains multiple or floating address lines. ADDRSCAN accepts up to six address lines in an input record and returns four address lines and a 10-byte ZIP Code field.
- Your input address record has more than two address lines where the street address is found.
- Your input city/state line floats.
- Apartment or floor information appears on a separate line from the street information. This information often provides a more precise assignment of the sector segment number.
- You want to select the USPS-preferred address type from a choice of addresses.
- You need to specify the format of the returned line information for your calling program to receive.
- Your ZIP Code is not fix-fielded, but you want to perform a ZIP Code lookup using a postal coding API such as Finalist. A fix-fielded ZIP Code is a ZIP Code that always begins in the same position of the same field within an address record. The next example illustrates address records for which the ZIP Code is not fix-fielded.

EXAMPLE

The examples below do *not* contain fix-fielded ZIP Codes. The ZIP Code can appear in a different location within each address record.

123 Main St
Roseville MI 48066

451 S Washington Ave
Apartment 1B
San Francisco, CA 94101

300 Birch Lane
Glen Ellyn IL
60137

How Do I Install the ADDRSCAN Option?

The ADDRSCAN Option is installed as part of your standard Finalist installation. The distribution media sent to you in your release package contains all data needed to perform ADDRSCAN processing.

How Does ADDRSCAN Processing Work?

ADDRSCAN processing includes these steps:

- Identification Section
- Concatenation Section
- Standardized Lines Section
- Returned Lines Section

1. Identify the address lines.

   Each address line passed to ADDRSCAN is identified as a firm line, street line, city line, urbanization line, or unknown line. After identifying each address line as a firm line, street line, city line, or unknown line, ADDRSCAN assigns each line a line type code. ADDRSCAN uses the line type code within the Concatenation Section to determine which lines can be combined to form standardized addresses. Table 1 describes the line type codes assigned during the identification process.

Table 1: Line Type Code Table

Line Type Code	Description
0	Firm line
1	PO box address line
2	Address line with both a range and a suffix word
3	Address line with a range but no suffix Address line with a suffix but no range
4	Rural route address line
5	Personal name Firm name (without firm words) Unidentified
6	Apartment type line
7	Possible city line
8	City line
9	Ignore this line
R	Rural route address line preceding a box line
B	Box address line following a rural route address line
M	Military address line

After identifying each address line and assigning a line type code, ADDRSCAN passes the lines for concatenation.

2. Concatenate (combine) related data from the six input address lines. These lines are then passed for further processing. The following concatenation rules determine the address line combination.

Table 2: Concatenation Rules

Rule Description
Line type code 6 lines append to line type code 2 lines.
Line type code 6 lines append to line type code 3 lines.
Line type code 8 lines append to line type code 7 lines.
Line type code B lines append to line type code R lines.

NOTE: ADDRSCAN combines lines only up to the point where the receiving line can hold all of the combined data. For example, assume there are three lines to combine, but the receiving line only has enough room for the first two lines. ADDRSCAN will combine only the first two lines and leave the existing third line.

a. Lines with a line type code 6, 7, or B append to the original street/city/rural line and form a new concatenated line.

b. The new concatenated line overlays the original street/city/rural line.

c. ADDRSCAN "reassigns" the line type code 6, 7, and B lines a new line type code of "9".

d. These lines are now part of the new concatenated line used to build the standardized address. In effect, the line type code "9" means "ignore this address line," and thereby prevents the lines from appearing twice in the standardized address. The following example demonstrates line type code assignment.

EXAMPLE

The address in this example is passed for the concatenation process. Note the line type codes that were assigned to each line during the identification process.

```
PO Box 123                      ** 1 **
123 Main Street                 ** 2 **
Apt 12A                         ** 6 **
Roseville MI 48066              ** 8 **
```

This example complies with the concatenation rules above, because the third line (line type code 6) will be appended to the second line (line type code 2). The combined address lines are ready for standardization. The next step describes the standardization process.

3. Standardize the address lines. Standardized lines are the original lines from your input file which are corrected and returned to the user input area, replacing your original input lines. The ADDRSCAN standardization process steps are:

   a. Return the standardized lines in any order. ADDRSCAN places the standardized lines in the order identified by codes passed from the calling program. If you enter "N" instead of valid codes, the system returns the lines in the order entered.

   b. Return codes to identify the standardized line types. ADDRSCAN returns line type codes that allow you to identify each of the standardized lines. Table 3 describes the standardized line type codes.

Table 3: Standardized Line Type Codes

Line Type Code	Description
0	Firm
1	P. O. box
2	Street (with suffix)
3	Street (without suffix)
4	Rural route
5	Personal name (probable)
8	City/state/ZIP Code
9	Miscellaneous (i.e., C/O)
M	Military
P	Puerto Rico URB Data

4. Return the processed lines. The Returned Lines section processing steps are:

   a. The standardized address is built in the order requested. If no order is requested, the lines are returned in the order entered.

   b. The ZIP + 4 Code is identified.

   c. The ZIP + 4 Code is removed from the returned city line and placed in a fixed field.

   NOTE: ADDRSCAN only identifies the ZIP + 4 Code during this process. The Finalist API will verify the ZIP + 4 Code.

   d. Two four-byte line indexes help you relate your returned address to your original input record and to the standardized address. Index1 contains the line numbers in your original input record which correspond to the returned lines. Index2 contains the line numbers in the standardized address which correspond to the returned lines.

Table 4: Index Pointer Table

Position	Description
1	Source of Returned Line 1
2	Source of Returned Line 2
3	Source of Puerto Rico urbanization data (Index1 only)
4	Source of Returned City Line

NOTE: The ZIP Code is returned in a separate field and does not display in the returned address lines.

   e. The Returned Lines Section builds the Returned Lines Address. The fourth, and last, step is to build the returned lines address. The returned lines section takes the standardized address, applies the format you specified for the Returned Line Options, and produces your returned lines address.

ADDRSCAN returns four 70-byte lines. The contents of each 70-byte line is shown in the next figure.

Table 5: Returned Lines Contents

Line Number	Description
Line 1	Street address data
Line 2	Street address data
Line 3	City and state
Line 4	Firm name (if supplied in the input record)

f. The four-line address passes back to the calling program in the format you requested. For additional examples of how return line addresses are built, refer to the section "ADDRSCAN Examples" later in this chapter.

EXAMPLE

The Returned Line Options are set to "56N". The figure shown next explains this setting.

Table 6: Returned Line Options Example

Parameter Position	Description
Return Line 1	5=Return first valid line above city line into Return Line 1
Return Line 2	6=Return second valid line above city line into Return Line 2
Return Line 3	N=No special options used
Return Line 4	Remains blank - no firm line present

ADDRSCAN Output

The following list describes the output generated during ADDRSCAN processing

- The original input returns left-justified and with only one blank between words.
- Valid special characters are permitted:
  &
  -
  /
  ' (single quote character)
  .
  :
  \
  ,
  #
  , (comma)

  NOTE: The pound sign (#) is the only special character you can use to begin a word. All lower case alpha characters translate to upper case. All other special characters are blanked out.

- ADDRSCAN allows words such as APT, Apartment, Suite, or Condo, to be the first words in a line when identifying an apartment type line. FLOOR, FL, or FLR can be the first or last word. Apartment type lines can appear anywhere and combine with the street address line. Up to two apartment type lines will be combined with the address line.
- ADDRSCAN treats a one-word line as an apartment type line if the word is one of the following:
  - All numeric (12345)
  - A number that ends with a single alpha (5B)
  - A number that starts with a single alpha (C5)
  - A token that starts with an alpha, followed by a hyphen (-) followed by a number (D-5). By default ADDRSCAN automatically concatenates these tokens to the address line.
- ZIP + 4 Codes are always returned, with a hyphen, in the ZIP Code field.
- If names, such as company and personal names, pass with the address line, ADDRSCAN finds and returns the valid address and city/state lines.
- Even if the street address and city lines are not in the proper order, ADDRSCAN returns the valid address and city/state lines. However, within the input lines the ZIP Code must follow the city/state data with a space between state and ZIP Code.
- The ADDRSCAN function includes tables that identify apartment, box, rural route, street, firms, and some Spanish words. These tables are used to identify the address lines.

Calling ADDRSCAN

You can call ADDRSCAN from a user-written driver. Sample C and COBOL programs are provided to assist you in this effort. Table 7 describes the files provided to support ADDRSCAN.

Table 7: Finalist ADDRSCAN Files

File	Description
addr021d.cob	COBOL copybook for calling ADDRSCAN
ADDRSCAN	z/OS load module
addrscan.h	Contains ADDRSCAN definitions
addrscan.dll	Finalist ADDRSCAN Windows .dll
addrscan.lib	Finalist ADDRSCAN Windows library
addrscan.a	Finalist ADDRSCAN UNIX library

Returned Line Options (ADDRPASS-OPTIONS)

You can set the returned line options to specify the format of the lines returned for address line 1 and address line 2. You can also specify a special option, such as separating combined address lines. The following sections describe the returned line options.

System Default

The system default for the returned line options is "95N". The figure shown next describes this setting.

Table 8: Default Returned Line Settings

Returned Line	Value	Description
1	9	Best street or box line from top. Otherwise, use the first firm or rural route line from the top.
2	5	First valid line above city line.
Special Option	N	No special options.

To use the default value, pass "YES" for the Returned Line Options.

User-Specified Formats

To specify the format of your returned lines address, select three options from the following tables (one for each returned line and a special option).

NOTE: When you enter options, you must enter all three options or the results will be undesirable.

To specify a user-defined format for Returned Line 1 (ADDRPASS-OPTION1), select one of the options shown in Table 9.

Table 9: Returned Line 1 Table

Code	Returned Line 1 Description
1	First valid line from top
3	First firm line from top
5	First valid line above city line
7	Street address line above city line
8	PO box or RR/HC line above city line
9	Best street or box line from top; if not found, the first firm or rural route line from top. When Option 9 is specified for returned line 1, ADDRSCAN performs selection from the top down as shown below: 1) First box line or complete address Example: PO BOX 12 or 123 Main St. 2) First street line with a range but no suffix Example: 123 Main First street line with a suffix but no range Example: Main St. 3) First rural route line Example: RR 1 Box 12 4) First firm type Example: Pitney Bowes

To specify a user-defined format for Returned Line 2 (ADDRPASS-OPTION2), select one of the options shown in Table 10.

Table 10: Returned Line 2 Table

Code	Returned Line 2 Description
B	Blank line
1	First valid line from top
2	Second valid line from top
3	First firm line from top
4	Second half of combined address line if first half is returned in line 1
5	First valid line above city line
6	Second valid line above city line
7	Street address line above city line
8	PO box or RR/HC line above city line

NOTE: If you make the same selection for Returned Line 1 and Returned Line 2, the second returned line will be blank.

To specify a special option (ADDRPASS-OPTION3), select one of the options in Table 11.

Table 11: Special Options Table

Code	Special Option Description
A	All special options
N	No special options
R	Add a range of "0" to street lines without a range Example: Main St becomes 0 Main St
S	Separate combined address lines

EXAMPLE

The first example below shows a combined address line. The second example shows the same address line using special option "S" (separate combined address lines).

```
123 MAIN ST RR 1 BOX 2
ROSEVILLE MI 48066

123 MAIN ST
RR 1 BOX 2
ROSEVILLE MI 48066
```

Returned Line Order (ADDRPASS-RTN-ORDER)

Set the Returned Line Order to specify the order in which the standardized lines are returned. Table 12 describes the available Returned Line Order codes.

Table 12: Return Line Order

Code	Description
0	Firm line
1	PO Box address line
2	Address line with both a range and a suffix word
3	Address line with a range but no suffix, Address line with a suffix but no range
4	Rural route address line
5	Personal name, Firm name (w/o firm words), Unidentified
6	Apartment type line
7	Possible city line
8	City line
9	Ignore this line
R	Rural route address line preceding a box line
B	Box address line following a rural route address line
M	Military address line

An N in the first return position says to return the default that is best address line 1, best address line 2, city state, firm. URB and ZIP are returned in separate fields.

EXAMPLE

This example demonstrates how you can specify the returned line order for a multi-line address in order to affect what information is returned.

```
Input Address:           Mrs P. R. Farber
                         Office World Inc
                         123 Main
                         4064 Monroe Street
                         RR 1 Box 12345
                         Toledo OH 43060

Returned Line Options: 35N

Returned Line Order:   0543218

Standardized Address:    OFFICE WORLD INC
                         MRS P. R. FARBER
                         RR 1 BOX 12345
                         123 MAIN
                         4064 MONROE STREET
                         TOLEDO OH 43060

Returned Address:      OFFICE WORLD INC
                       4064 MONROE STREET
                       TOLEDO OH

Returned ZIP Code:   43060

Returned Line Type Code:  054328
```

Concatenation Line Maximum (ADDRPASS-LNMX-NUM)

Concatenation occurs when related data is located on multiple lines.

```
123 MAIN ST
BLDG 4
APT 12.
```

ADDRSCAN will concatenate the previous lines as shown next.

```
123 MAIN ST BLDG 4 APT 12
```

Note that the concatenation line maximum is typically set to 70 characters. This may cause problems if you are going to use the standardized lines for reuse in your records and your original records are not large enough to hold the new line. In such cases, you can change the maximum standardized line length to prevent line overflow.

Processing Options (ADDRPASS-TYPES)

To prevent ADDRSCAN from concatenating lines beginning with a # to existing address lines, enter a # in one of the first three columns of ADDRPASS-TYPES.

To prevent ADDRSCAN from recognizing periods as a valid characters (to specify that a period should be removed before scanning address lines), enter a P in one of the first three columns of ADDRPASS-TYPES.

To prevent ADDRSCAN from concatenating one-word lines to address lines, enter a D in one of the first three columns of ADDRPASS-TYPES.

NOTE: Since the Processing Options are using a field that is also an output field (ADDRPASS-TYPES), Processing Options must be reset before each call to ADDRSCAN.

CALLAREA Structure Definition

The definition of the CALLAREA structure that is used to communicate with the ADDRSCAN function is contained in the header file addrscan.h as shown below.

```
#define LN  70
.
.
#define NUM_INLINE     6
#define NUM_OUTLINE    4
#define NUM_OPTS       3
#define INDEX_SIZE     4
.
.
typedef struct {
   char  pInline[NUM_INLINE][LN];
   char  pOutline[NUM_OUTLINE][LN];
   char  urb[URB_SIZE];
   char  pAddrNdx1[INDEX_SIZE];
   char  pAddrNdx2[INDEX_SIZE];
   char  pAddrZip4[10];
   char  pAddrOpts[NUM_OPTS];
   char  pAddrTyps[6];
   char  pAddrRord[9];
   SHORT sAddrLnMx;
   char  University;
   char  Dormatory;
   char  pAddrAnch[4];
} CALLAREA, *PCALLAREA;
```

Table 13 contains a description for each CALLAREA structure member.

Table 13: CALLAREA Member Descriptions (Part 1 of 2)

Member Name	Length	Description
char pInline[0]	70	Input address line 1.
char pInline[1]	70	Input address line 2.
char pInline[2]	70	Input address line 3.
char pInline[3]	70	Input address line 4.
char pInline[4]	70	Input address line 5.
char pInline[5]	70	Input address line 6.
char pOutline[0]	70	Returned address line 1.
char pOutline[1]	70	Returned address line 2.
char pOutline[2]	70	Returned city/state line.
char pOutline[3]	70	Returned firm line (if a firm was provided on input).
char urb	30	Returned urbanization line.
char pAddrNdx1	4	Returned index containing the line numbers of your original input address which correspond to the returned address lines.

Table 13: CALLAREA Member Descriptions (Part 2 of 2)

Member Name	Length	Description
char pAddrNdx2	4	Returned index containing the line numbers of the standardized input address which correspond to the returned address lines.
char pAddrZip4	10	Returned ZIP + 4 Code. Consists of the 5-digit ZIP Code, a hyphen, and the sector/segment.
char pAddrOpts	3	Returned Line Options (each 1 character long). Defines the option for selecting the line to be returned in output line 1 (pOutline[0]), the option for selecting the line to be returned in output line 2 (pOutline[1]), and the special option selector. **NOTE:** If pAddrOpts contains HIGH-VALUES (0xFFFFFF), ADDRSCAN terminates and frees all acquired storage.
char pAddrTyps	6	Returned line type codes that allow you to identify each of the standardized input lines. pAddrTyps can also be used as an input field to alter ADDRSCAN processing. To prevent ADDRSCAN from concatenating lines beginning with # to existing address lines, enter # in the pAddrTyps field. To prevent ADDRSCAN from recognizing periods as valid characters and to specify that periods should be removed before scanning address lines, enter a P in the pAddrTyps field. To prevent ADDRSCAN from concatenating a one-word line to an address line, enter a D in the pAddrTyps field. Note that, If an address line ends with a Unit Designator, a one-word will always be concatenated. **NOTE:** The values #, P, and D can populate any of the first three positions of the pAddrTyps field.
char pAddrRord	9	Returned Line Order. Defines the order in which standardized lines are returned.
short sAddrLnMx	2	Concatenation Line Maximum. Defines the maximum length of the standardized input lines.
char University	1	Set to "U" if university word found.
char Dormatory	1	Set to "D" if dormitory word found.
char pAddrAnch	4	Set to HIGH-VALUES (0xFF) prior to the first call.

COBOL Copybook Version

```
*****************************************************************
*          *          ADDRSCAN CALL-AREA          *             *
*          ************************************** *             *
* NOTE:  THE INITIAL VALUE OF ADDRPASS-ANCHOR WILL BE IGNORED   *
*        BY THE COMPILER IF THIS IS IN THE FILE SECTION OR      *
*        LINKAGE SECTION.  IN THAT CASE, MOVE -1 TO             *
*        ADDRPASS-ANCH-NUM BEFORE THE FIRST BATCH ADDRSCAN      *
*        CALL BUT NOT ON SUBSEQUENT CALLS.  IN CICS, THE VALUE  *
*        DOES NOT NEED TO BE INITIALIZED AT ALL.                *
*****************************************************************
 01  ADDRPASS-CALL-AREA.
     05  ADDRPASS-PASSED-LINES.
         10  ADDRPASS-INLINE1   PIC X(70).
         10  ADDRPASS-INLINE2   PIC X(70).
         10  ADDRPASS-INLINE3   PIC X(70).
         10  ADDRPASS-INLINE4   PIC X(70).
         10  ADDRPASS-INLINE5   PIC X(70).
         10  ADDRPASS-INLINE6   PIC X(70).
     05  ADDRPASS-RETURNED-LINES.
         10  ADDRPASS-OUTLINE1  PIC X(70).
         10  ADDRPASS-OUTLINE2  PIC X(70).
         10  ADDRPASS-OUTLINE3  PIC X(70).
         10  ADDRPASS-OUTLINE4  PIC X(70).
         10  ADDRPASS-OUTURB    PIC X(30).
     05  ADDRPASS-INDEXES.
         10  ADDRPASS-INDEX1    PIC X(04).
         10  ADDRPASS-INDEX2    PIC X(04).
     05  ADDRPASS-ZIP-PLUS-4.
         10  ADDRPASS-ZIP5      PIC X(05).
         10  ADDRPASS-HYPHEN    PIC X(01).
         10  ADDRPASS-SECSEG    PIC X(04).
     05  ADDRPASS-OPTIONS.
         10  ADDRPASS-OPTION1   PIC X(01) VALUE '9'.
         10  ADDRPASS-OPTION2   PIC X(01) VALUE 'B'.
         10  ADDRPASS-OPTION3   PIC X(01) VALUE 'N'.
     05  ADDRPASS-TYPES.
         10  ADDRPASS-TYPE1     PIC X(01).
         10  ADDRPASS-TYPE2     PIC X(01).
         10  ADDRPASS-TYPE3     PIC X(01).
         10  ADDRPASS-TYPE4     PIC X(01).
         10  ADDRPASS-TYPE5     PIC X(01).
         10  ADDRPASS-TYPE6     PIC X(01).
     05  ADDRPASS-RTN-ORDER.
         10  ADDRPASS-RTN-ORD1  PIC X(01) VALUE 'N'.
         10  ADDRPASS-RTN-ORD2  PIC X(01).
         10  ADDRPASS-RTN-ORD3  PIC X(01).
         10  ADDRPASS-RTN-ORD4  PIC X(01).
         10  ADDRPASS-RTN-ORD5  PIC X(01).
         10  ADDRPASS-RTN-ORD6  PIC X(01).
         10  ADDRPASS-RTN-ORD7  PIC X(01).
         10  ADDRPASS-RTN-ORD8  PIC X(01).
         10  ADDRPASS-RTN-ORD9  PIC X(01).
     05  ADDRPASS-LINE-MAX.
         10  ADDRPASS-LNMX-NUM  PIC 9(04) COMP VALUE 70.
         10  ADDRPASS-UNIV-WORD PIC X(01) VALUE SPACES.
         10  ADDRPASS-DORM-WORD PIC X(01) VALUE SPACES.
    05  ADDRPASS-ANCHOR.
         10  ADDRPASS-ANCH-NUM  PIC S9(8) COMP VALUE -1.
**************   END OF ADDRSCAN CALL-AREA   ******************
```

The following examples illustrate the values you can return within the ZIP Code field pAddrZip4.

EXAMPLE

ADDRSCAN returns a ZIP Code field consisting of a five-digit ZIP Code, a hyphen, and a sector segment number. The hyphen separates the ZIP Code from the sector segment number as shown next.

```
60148-6494
```

EXAMPLE

If you do not have a sector segment number present in the record, the last five positions will be blank. If ADDRSCAN is unable to locate a ZIP Code within your address data, the ZIP Code field returns as shown next.

```
00000-0000
```

ADDRTBLS

The ADDRTBLS module provides a method for setting up exceptions for ADDRSCAN processing. For example, you might want to use the ADDRTBLS module to set up exceptions for ADDRSCAN processing to process:

- P.O. SMITH as a name instead of as post office box
- VISTA SANTA ROSA as a city name instead of as an URB
- STAR RANCH as a city instead of an alias for STAR ROUTE

ADDRTBLS is provided as BAL source in the FNSOURCE library. Follow these steps for using ADDRTBLS.

1. Modify the ADDRTBLS source
2. Assemble ADDRTBLS and link into a load module. Link the load module using AMODE=31, RMODE=ANY.
3. Place the load module into the Finalist batch, CICS, and IMS load libraries.

NOTE: ADDRTBLS is only available on the mainframe platform. ADDRTBLS is not available on other platforms.

Calling ADDRSCAN With the Finalist Driver

You can call ADDRSCAN using the Finalist driver by specifying the cAddrScan=YES option in your JOB file. You also use the same cAddrScan parameter to specify ADDRSCAN configuration options. For example, cAddrScan=56N.

Calling ADDRSCAN With A User-Written Driver

The steps to call ADDRSCAN with a user-written driver are described next.

1. Define a local CALLAREA structure.
2. Initialize the entire CALLAREA structure to blanks except for pAddrAnch (ADDRPASS-ANCH-NUM) which should be HIGH-VALUES.
3. Set any members of the CALLAREA structure that you want to remain the same from call to call (i.e., the maximum length of the standardized address lines).
4. Initialize the input address lines of the CALLAREA structure to blanks (also do this before each subsequent call to ADDRSCAN).
5. Move the data from your input record to the appropriate members within the CALLAREA structure.
6. Call the ADDRSCAN function, passing the address of the CALLAREA structure.
7. Perform any custom processing using the returned address lines, returned city/state line, and returned ZIP Code field within the CALLAREA structure.
8. After ADDRSCAN processing, set pAddrOpts (ADDRPASS-OPTIONS) to HIGH-VALUES to terminate ADDRSCAN and make one last call.

ADDRSCAN Examples

The following table provides you with samples of how ADDRSCAN processes input records and builds the standardized lines, returned lines, and the line indexes. For our examples, option "54S" was specified for the Returned Line Options.

Input Address	Standardized Address	Returned Lines Address
PO Box 123 123 Main St Apt 12A RoseVille Mi 48066	PO BOX 123 123 MAIN ST APT 12A ROSEVILLE MI 48066	123 MAIN ST APT 12A ROSEVILLE MI
Index = 2004	**Index2 = 2003**	**ZIP Code = 48066**

Input Address	Standardized Address	Returned Lines Address
123 Main St Box 12A Roseville MI 48066	123 MAIN ST BOX 12A ROSEVILLE MI 48066	123 MAIN ST BOX 12A ROSEVILLE MI
Index = 1102	**Index2 = 1203**	**ZIP Code = 48066**

Input Address	Standardized Address	Returned Lines Address
123 Main St RR 1 Box 123 Roseville Mi 48066	123 MAIN ST RR 1 BOX 123 ROSEVILLE MI 48066	123 MAIN ST RR 1 BOX 123 ROSEVILLE MI
Index = 1102	**Index2 = 1203**	**ZIP Code = 48066**

Input Address	Standardized Address	Returned Lines Address
123 Main St 2nd Fl Roseville Mi 48066	123 MAIN ST 2ND FL ROSEVILLE MI 48066	123 MAIN ST 2ND FL ROSEVILLE MI
Index = 1004	**Index2 = 1002**	**ZIP Code = 48066**

Input Address	Standardized Address	Returned Lines Address
C/O John Doe 123 Main St Roseville Mi 48066	C/O JOHN DOE 123 MAIN ST ROSEVILLE MI 48066	123 MAIN ST ROSEVILLE MI
Index = 2003	**Index2 = 2003**	**ZIP Code = 48066**

Input Address	Standardized Address	Returned Lines Address
RR1 Box 123 Roseville Mi 48066	RR1 BOX 123 ROSEVILLE MI 48066	RR1 BOX 123 ROSEVILLE MI
Index = 1003	**Index2 = 1002**	**ZIP Code = 48066**

Input Address	Standardized Address	Returned Lines Address
1 Elm St Apt 6 Bldg 1 Roseville Mi 48066	1 ELM ST APT 6 BLDG 1 ROSEVILLE MI 48066	1 ELM ST APT 6 BLDG 1 ROSEVILLE MI
Index = 1004	**Index2 = 1002**	**ZIP Code = 48066**

Input Address	Standardized Address	Returned Lines Address
25 Calle 3 Urb San Juan Bautista Barceloneta PR 00607	25 CALLE 3 URB SAN JUAN BAUTISTA BARCELONETA PR 00607	25 CALLE 3 BARCELONETA PR
Index = 1023	**Index2 = 1003**	**ZIP Code = 00607**

C Driver Example

A C program listing is shown next.

```
/*************************************/
/*  ADDRSCAN sample program          */
/*************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "pbfnxdef.h"
#include "addrscan.h"

void addrscan(PCALLAREA pCallArea);

void main( int argc, char *argv[] );
void main( int argc, char *argv[] )
{
   CALLAREA  CallArea;
   short     i;
   char      pLine[ NUM_INLINE ][ LN+1 ];
   INTEGER   nDone;

   /* initialize CallArea with blank space
    * do NOT use '\0' char.
    */
   memset(&CallArea, ' ', sizeof (CALLAREA));
   memset(CallArea.pAddrAnch, 0xFF, sizeof(CallArea.pAddrAnch));

   /* Fill the input lines with data. */
   strcpy( pLine[0], "pbss corp." );
   strcpy( pLine[1], "123 MAIN street" );
   strcpy( pLine[2], "APT 12A" );
   strcpy( pLine[3], "Roseville Mi" );
   strcpy( pLine[4], "48066" );
   strcpy( pLine[5], "" );
   /* Set the maximum length of the standardized lines. */
   CallArea.sAddrLnMx = LL;

   /* Simulate a processing loop because a real application would */
   /* call addrscan for each of many records.                     */
   nDone = 0;
   while ( !nDone )

      {
      /* Fill the input lines with blank characters. */
      memset( CallArea.pInline[0], (char) ' ', LL * NUM_INLINE );

      /* Copy the input lines into the CallArea structure. */
      for ( i = 0 ; i < NUM_INLINE ; i++ )
         {
         memcpy( CallArea.pInline[i], pLine[i], strlen( pLine[i] ) );
         }

      /* Set the format of the returned address lines to "54S". */
      memcpy( CallArea.pAddrOpts, "54S", 3 );

      /* Specify no special return order. */
      memcpy( CallArea.pAddrRord, "N", 1 );

      printf( "\nINPUT LINES:\n\n" );
      for ( i = 0; i < NUM_INLINE ; i++ )
         {
         printf( "Line #%1d : [%40.40s]\n", i+1, CallArea.pInline[i] );
         }

      addrscan( &CallArea );

      printf( "\nSTANDARDIZED INPUT LINES:\n\n" );
      for ( i = 0; i < NUM_INLINE ; i++ )
         {
         printf( "Line #%1d : [%40.40s]\n", i+1, CallArea.pInline[i] );
         }

      printf ("\nRETURNED INFORMATION:\n\n");
      for ( i = 0 ; i < NUM_OUTLINE ; i++ )
```

```
            {
            printf( "Line #%1d : [%40.40s]\n", i+1, CallArea.pOutline[i] );
            }

         printf( "Index1  : [%4.4s]\n",   CallArea.pAddrNdx1 );
         printf( "Index2  : [%4.4s]\n",   CallArea.pAddrNdx2 );
         printf( "Zip4    : [%10.10s]\n", CallArea.pAddrZip4 );
         printf( "Types   : [%6.6s]\n",   CallArea.pAddrTyps );

         /* Cause the loop to be exited. */
         nDone = 1;

      } /* while ( !nDone ) */

      fflush( stdout );

   } /* End of "main" */
```

Figure 1: C Program Listing Example

COBOL Driver Example

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. 'ADDRTEST'.
 DATE-COMPILED.
** ============================================================
*
*
*  AUTHOR.
*     PITNEY BOWES SOFTWARE
*     ALL RIGHTS RESERVED.
*
*     PROGRAMS AND DESIGN (C) COPYRIGHT 2010
*
** ============================================================
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER. IBM-3270 WITH DEBUGGING MODE.
*OBJECT-COMPUTER. IBM-3270.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
*
* PROGRAM = ADDRTEST
*
* PROGRAM TYPE = BATCH DRIVER
*
* APPLICATION = FINALIST
*
* CALLED PROGRAMS:
*     ADDRSCAN - AN EXTERNAL ROUTINE FOR IDENTIFYING
*    ADDRESS LINES
*
* INPUTS:
*     NONE
*
* OUTPUTS:
*     NONE
*
* PROGRAM NARRATIVE:
*     ADDRTEST IS TEST DRIVER PROGRAM.
*     ITS PURPOSE IS TO DEMONSTRATE HOW TO CALL ADDRSCAN.
*
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*
 DATA DIVISION.
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*

*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*
 FILE SECTION.
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*

*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*
 WORKING-STORAGE SECTION.
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*

 77  FILLER     PIC X(45)   VALUE
     '** WORKING STORAGE OF ADDRTEST STARTS HERE **'.

* ADDRPASS-CALL-AREA
     COPY ADDR021D.

 77  FILLER    PIC X(43)   VALUE
     '** WORKING STORAGE OF ADDRTEST ENDS HERE **'.

*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*
 LINKAGE SECTION.
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*

*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*
 PROCEDURE DIVISION.
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*
 0000-MAIN-LOGIC.
*
     INITIALIZE ADDRPASS-CALL-AREA.
*
     MOVE '95N' TO ADDRPASS-OPTIONS
```

```
     MOVE HIGH-VALUES TO ADDRPASS-ANCHOR
     MOVE 'N' TO ADDRPASS-RTN-ORD1
     MOVE 70 TO ADDRPASS-LNMX-NUM
*
     MOVE 'PITNEY BOWES'    TO ADDRPASS-INLINE1
     MOVE '2200 WESTERN CT' TO ADDRPASS-INLINE2
     MOVE 'STE 100'   TO ADDRPASS-INLINE3
     MOVE 'LISLE'     TO ADDRPASS-INLINE4
     MOVE 'IL'  TO ADDRPASS-INLINE5
     MOVE '60532-3618'TO ADDRPASS-INLINE6
*
* INPUT DATA
*
     DISPLAY 'ADDRSCAN INPUT VALUES'
     DISPLAY 'ADDROPTS:' ADDRPASS-OPTIONS
     DISPLAY 'ADDRRORD:' ADDRPASS-RTN-ORDER
     DISPLAY 'ADDRTYPS:' ADDRPASS-TYPES
     DISPLAY 'INLINE1 :' ADDRPASS-INLINE1
     DISPLAY 'INLINE2 :' ADDRPASS-INLINE2
     DISPLAY 'INLINE3 :' ADDRPASS-INLINE3
     DISPLAY 'INLINE4 :' ADDRPASS-INLINE4
     DISPLAY 'INLINE5 :' ADDRPASS-INLINE5
     DISPLAY 'INLINE6 :' ADDRPASS-INLINE6
     DISPLAY ' '
*
     CALL 'ADDRSCAN'   USING
    BY REFERENCE ADDRPASS-CALL-AREA
     DISPLAY 'ADDRSCAN OUTPUT VALUES'
     DISPLAY 'OUTLINE1:' ADDRPASS-OUTLINE1
     DISPLAY 'OUTLINE2:' ADDRPASS-OUTLINE2
     DISPLAY 'OUTLINE3(CS)  :' ADDRPASS-OUTLINE3
     DISPLAY 'OUTLINE4(FIRM):' ADDRPASS-OUTLINE4
     DISPLAY 'ADDRNDX1:' ADDRPASS-INDEX1
     DISPLAY 'ADDRNDX2:' ADDRPASS-INDEX2
     DISPLAY 'ZIP-ZIP4:' ADDRPASS-ZIP-PLUS-4
     DISPLAY 'UNIVERSITY    :' ADDRPASS-UNIV-WORD
     DISPLAY 'DORMATORY     :' ADDRPASS-DORM-WORD
     DISPLAY ' '
*
     GOBACK.
*
 0000-EXIT.
     EXIT.
 END PROGRAM 'ADDRTEST'.
```

CHAPTER 11

Residential Delivery Indicator (RDI) Option

This chapter describes the Finalist Residential Delivery Indicator (RDI) Option. You can use the Residential Delivery Indicator (RDI) Option to determine whether an address is a residential or business address.

What is the Residential Delivery Indicator (RDI) Option?

The USPS provides a method for determining if a mailing address is a residential or business address.

For example, if a person runs a small business out of their home, RDI would identify that address as a residential address and not a business address. However, if the address is flagged as not residential, then a business is known to operate at this location.

Can I Use the Residential Delivery Indicator (RDI) Option?

The Residential Delivery Indicator (RDI) Option is available to all Finalist customers.

Why Should I Use the Residential Delivery Indicator (RDI) Option?

If you would like to differentiate residential mailing from business mailing, then you should use the Residential Delivery Indicator (RDI) Option.

How Do I Install the Residential Delivery Indicator (RDI) Option?

The Residential Delivery Indicator (RDI) Option is installed as part of your standard Finalist installation. After installing Finalist, you must activate the Residential Delivery Indicator (RDI) Option to perform RDI processing.

Pitney Bowes Software does not distribute the RDI databases. You must contact the USPS to obtain RDI databases.

NOTE: For Windows and UNIX, use the RDI databases as provided by the USPS. Mainframe users must run special JCL (RDREPROD) to make the RDI databases available to Finalist on the mainframe. Mainframe users can access RDI in batch, CICS, and IMS environments.

How Do I Activate Residential Delivery Indicator (RDI) Processing?

To activate RDI processing, you will need to define the fields listed below.

- **RDI Filepath** — Define the path to the RDI files (Windows and UNIX users only).
- **Residential Delivery Indicator** — This field indicates whether to perform Residential Delivery Indicator (RDI) processing.

Methods for Activating RDI Processing

To activate RDI processing, use one of the following methods.

- **pbfn.cfg Configuration File** — If you prefer to configure your Finalist installation using global settings, you can define the RDI fields in your pbfn.cfg configuration file. For more information on the pbfn.cfg file, refer to your *Working With Finalist Guide*.
- **PBFNSetupDef Structure** — If you prefer to configure your Finalist installation using the Finalist structures, you can define the RDI fields in the PBFNSetupDef structure. For more information on the PBFNSetupDef structure, refer to your *Finalist Developer's Reference Guide*.
- **Workbench or Lookup Tool** — If you prefer to configure your Finalist installation using the Finalist Workbench or Lookup Tool GUI screens, you can define the RDI field in the **Product Tab** on the *PBFN Config Setting* dialog box. For more information on the Finalist Workbench and Lookup Tool, refer to your *Working With Finalist Guide*.
- **Compatibility Interface (CI)** — If you are using the Finalist Compatibility Interface (CI), you can specify RDI processing when you call Finalist. For more information on the Compatibility Interface (CI), refer to Chapter 3, Using the Compatibility Interface in your *Finalist Developer's Reference Guide*.

Each method for activating RDI processing is described in the following sections.

Using pbfn.cfg to Activate RDI Processing

To activate RDI processing using the pbfn.cfg file, complete the following fields in your pbfn.cfg file.

Table 1: pbfn.cfg File — RDI Settings

pbfn.cfg Field	Description
RDI Filepath	Specify the Residential Delivery Indicator (RDI) file name and path.
Residential Delivery Indicator	Indicate whether to perform Residential Delivery Indicator (RDI) processing: • **OFF** — Do not perform Residential Delivery Indicator (RDI) processing. • **ON** — Perform Residential Delivery Indicator (RDI) processing. • blank — Defaults to **OFF**.

Using the PBFNSetupDef Structure to Activate RDI Processing

To activate RDI processing using the PBFNSetupDef structure, complete the following fields in the PBFNSetupDef structure.

Table 2: PBFNSetupDef Structure — RDI Settings

PBFNSetupDef Field	Description
RDIFilePath	Specify the Residential Delivery Indicator (RDI) file path.
cAssignRDI	Indicate whether to perform Residential Delivery Indicator (RDI) processing: • **OFF** — Do not perform Residential Delivery Indicator (RDI) processing. • **ON** — Perform Residential Delivery Indicator (RDI) processing. • blank — Defaults to **OFF**.

Using the Finalist Workbench to Activate RDI Processing

To activate RDI processing using the Finalist Workbench:

1. Launch the Finalist Workbench.
2. From the **Tools** menu, select **PBFN Setup**.
3. Select the **Product** tab on the *PBFN Config Setting* dialog box.

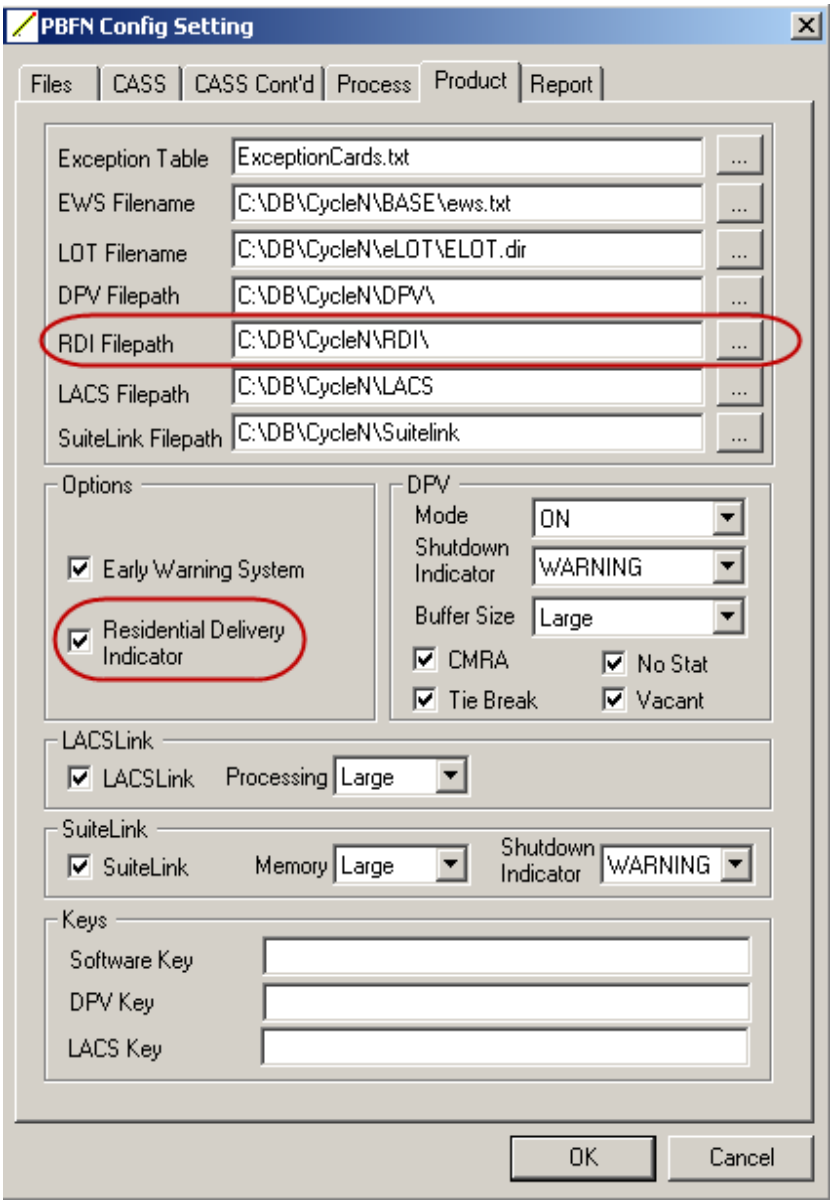

Figure 1: PBFN Config Setting Dialog Box — Product Tab

4. Complete the following fields on the **Product** tab.

Table 3: PBFN Config Setting Dialog Box — Product Tab

Field	Description
RDI Filepath	Specify the Residential Delivery Indicator (RDI) file path or click on the **Browse** button (located to the right of the field) to open a dialog box and select the path.
Residential Delivery Indicator	Check the Residential Delivery Indicator box to perform Residential Delivery Indicator (RDI) processing.

5. Click **OK** to save your settings.

Using the Compatibility Interface (CI) to Activate RDI Processing

To activate RDI™ processing for the Compatibility Interface (CI), specify the following when calling the CI:

```
05  FINAL-RDI-OPT              PIC  X(01).
    88  FINALIST-RDI-OFF            VALUE 'N'.
    88  FINALIST-RDI-ON             VALUE 'Y'.
```

How Does RDI Processing Work?

Finalist performs RDI processing as described below.

1. During processing of your input file, Finalist validates the addresses.
2. Finalist looks up the coded address in the RDI databases.
3. RDI returns a flag, through Finalist, indicating if this is a residential address.

Structures Containing RDI Information

The structures listed below include data to facilitate RDI processing. For detailed information on these structures, refer to your *Finalist Developer's Reference Guide*.

- PBFNIMSSetupDef
- PBFNSetupDef (includes two RDI setup parameters)

RDI Output

The following sections describe Residential Delivery Indicator (RDI) Option output.

RDI Return Information

To post the Residential Delivery Indicator using Finalist.exe, define the following keyword in your Definition File.

Table 4: Definition File Layout Component Keyword — RDI™ Settings

Definition Keyword	Description
oRdi=x,y; a[,y]	Identifies the position (x) and length (y) of the output Residential Delivery Indicator.

The RDI™ output is returned in the PBFNParsedAdrAltDef (PBFNParsedAdrAltDef) and the PBFNProcessDataDef (PBFNProcessDataAltDef) structures in the following field.

Table 5: RDI Output Field

Field	Description
cRDI	Character defining the Residential Delivery Indicator. • **Y** — Address is a residential delivery. • **N** — Address is a business delivery. Blank — Failed address lookup (did not return a +4), or RDI was not active.

The CI returns the RDI output in the following field.

Table 6: Finalist Return Area - Function 4, 5, 6, or 7 Process Call

COBOL Field Name/ C Field Name	Description
FINAL-RDI-RETURN-CODE caRDIReturnCode	Contains a "Y" if the matched address is a residential delivery. Contains an "N" if the matched address is a business delivery. This field is blank if the address fails address lookup, or the RDI™ Option is not active.

RDI Information on Finalist Batch Report

The Finalist Batch Report provides statistics for RDI processing. For detailed information on the Finalist Batch Report, refer to your *Working With Finalist Guide*.

GLOSSARY

Term	Definition
3553 Report	USPS CASS Report.
API	Application programming interface. A set of routines, protocols, and tools for building software applications.
APO	Army Post Office. Mail for Army personnel is sent to one of several APOs in the United States. Each APO then forwards the mail to military bases throughout the world. Finalist does not process APOs as a conventional address. If Finalist does not find an exact match, the record will fail.
Barcode	An array of rectangular marks and spaces which appear in a predetermined pattern following unambiguous rules in a specific way to represent elements of data which are referred to as characters.
Base Street Name	The base street name is the street name that the USPS actually lists in the ZIP + 4 postal file. For CASS certification purposes, your processing job should return the base street name. However, the USPS will accept either the alias street address or the base street address on the mail piece.
Carrier Route Code	A four-position code that designates the appropriate delivery route for a particular address. The USPS establishes carrier route coding schemes. Each scheme is ZIP specific.
CASS	USPS Coding Accuracy Support System.
CD-ROM	CD-ROM stands for compact disk, read-only memory. The CD which looks the same as an audio CD can store about 650 megabytes of data. You cannot delete or update a file on a CD-ROM.
Check Character	A character included within a symbol with a value used for the purpose of performing a mathematical check to ensure the accuracy of the data.
Check Digit	See Check Character.
CICS	Customer Information Control System.

City Delivery A combination of delivery methods within a community where all residential and business customers are served according to postal regulations.

City Place Name The name of a city, place, town, or other name by which a five-digit ZIP Code is commonly known.

CMRA Private companies offering mailbox rental services to individuals and businesses are Commercial Mail Receiving Agents (CMRA). See also PMB.

Conventional Address Address in which there is a street name and number. The street direction indicator (if any) in a strictly conventional address is between the street range number and the street name. For example, 1710 N. FOREST is a typical address line from a conventional address.

Delimiter A character that marks the beginning or end of a unit of data.

Delivery Point Barcode (DPBC) A 14-digit barcode consisting of two framing characters, a five-byte ZIP Code, a four-byte +4 code, a two-byte delivery point, and a one-digit modulo check digit. Modulo is a term used to describe several packet-switched network parameters, such as packet number (i.e., set to modulo 10, counted from 0 to 9). When the maximum count is exceeded, the counter is reset to 0.

Delivery Point Validation (DPV) The Finalist Delivery Point Validation (DPV) Option uses DPV data available from the USPS to determine whether an address actually exists. The Delivery Point Validation (DPV) Option can verify the existence of an address to as fine a level as an apartment or suite. Mailers can use the Delivery Point Validation (DPV) Option to ensure the addresses in their address file are actual physical addresses to which the USPS delivers mail.

Deprecated A term applied to features that are superseded and should be avoided. Although deprecated features remain in the current version, the use of deprecated features generates warning messages. Deprecated features will be removed in the future. Features are deprecated in order to give programmers using the feature time to bring their code into compliance with the new standard.

Directional NE, West, etc.

DLL See Dynamic Link Library (DLL).

Dual Address A dual address is an address that contains more than one mailable address (i.e., an address that contains both a PO BOX and a street address).

DVD (or DVD-ROM) DVD-ROM stands for "Digital Versatile Disk" or "Digital Video Disk", read only memory. A DVD holds 4.7 gigabytes of data. You cannot delete or update a file on a DVD-ROM.

Dynamic Link Library (DLL) Dynamic Link Library called dynamically at execution time.

Early Warning System (EWS) USPS CASS 2002-2003 regulations require all CASS-certified software to be able to read the USPS Early Warning System (EWS) File. The Finalist Early Warning System (EWS) Option verifies input addresses that are not found in the current ZIP + 4 File against the USPS EWS File. If an input address is found in the EWS File, the input address is not matched to any similar addresses in the current ZIP + 4 File. Instead, the input address fails and is not coded until the ZIP + 4 File is updated with the correct address from the USPS EWS File.

False-Positive Violation The USPS has put in place security measures to ensure mailers using DPV and LACS^{Link} processing do not use these applications to generate mailing lists. If the USPS identifies a mailer as repetitively generating false-positive violations, the USPS may direct Pitney Bowes Software to invalidate their license. Towards that end, the USPS has created and monitors addresses that generate a false-positive result. The USPS requires Pitney Bowes Software to report any organization generating a false-positive result during DPV and/or LACS^{Link} processing.

Finance Number The Finance Number consists of a state code (first two digits) and a postal installation code. There is a unique Finance Number for each post office name.

FIPS Code Federal Information Processing Standards (FIPS) code. A FIPS code is a two-character state code followed by a three-character county code.

FPO Fleet Post Office. Mail for Navy personnel is sent to one of several FPOs in the United States. Each FPO then forwards the mail to Navy bases throughout the world. Finalist does not process FPOs as a conventional address. If Finalist does not find an exact match, the record will fail.

Frame Characters A special barcode character that provides the scanner with the start and stop instructions. Place the frame character at the beginning and ending of the Delivery Point barcode.

HCHighway Contract Route HC provides for the transportation of mail between post offices or other designated points where mail is received or distributed.

IMS Information Management System.

LACS^{Link} The USPS LACS^{Link} database contains data on address conversions.

Last Line Information The address last line information contains the city, state, and ZIP Code.

Line of Travel (LOT) Line of travel sequence is an option for mailers who prepare carrier route mailings other than high-density/125-piece or saturation mailings. LOT sequencing is required for Basic Enhanced Carrier Route Standard Mail except automation-compatible, letter-size pieces. LOT sequence is not an exact walk sequence but a sequence of ZIP + 4 Codes arranged in the order that the route is served by a carrier. First the ZIP + 4 groups are sequenced. Then the addresses within each are identified as being in ascending or descending order.

Mailing Statement A postal service form the mailer fills out, which lists the number of pieces of mail he or she is submitting at discount prices.

Modulo A term to describe the adjustment value to bring a number up to the next multiple of its base number. For example, 13 modulo 10 has a value of 7. That is, you have to add 7 to 13 to bring it up to the next multiple of 20.

Mother ZIP The term Mother ZIP is used to refer to the lowest ZIP Code within the finance area.

Nickname Alias Street Name An alternate street name, maintained at the 5-digit ZIP Code level. It could be a name by which a street was formerly known, or a commonly used nickname for the street.

Non-Deliverable Address Non-deliverable areas include vacant lots and land that borders railroad tracks, areas to which the USPS does not deliver mail.

Non-Mailing Name A city name that is recognized by the USPS, but is not the preferred name for the ZIP Code. This is often a vanity name for the area.

Non-Parsed Address Process Data structures) - Components of an address are combined into a single field. For example, an address 1 field might contain the Range, Street name, and Street suffix all separated by spaces. A last line address could consist of City, State, and/or ZIP all in the same field. With non-parsed address components, Finalist must spend additional time and resources to determine the individual components.

Parse To analyze or separate into component parts.

Parsed Address Components Components of an address are stored independently of each other. Components of a complete street address are some combination of: Range, Pre Directional, Street Name, Post Directional, Street Suffix, Unit Designator, Unit Range, Unit 2 Designator, Unit 2 Range, PMB Designator, PMB Range. Components of a last line are City, State, ZIP, ZIP + 4, Delivery Point, Carrier Route, Advanced Bar Code. With Parsed Address components, Finalist does not need to determine the individual components.

PMB A private mail box.

Point Of Entry The point (post office) from which mail is entered (submitted).

Post-Directional A geographic direction which follows the street name.

Pre-Directional A geographic direction which precedes the street name.

Preferred Alias Street Name Street names that are not standardized, that is, those addresses that include directional or suffix words as part of the street name, and not in their own fields. For example, a standardized address such as NE Military Sq would list NE in the pre-directional field, Military in the street name field and SQ in the suffix field. In contrast, the preferred alias street name would list Square in its non-standardized for as part of the street name (i.e., Military Square).

Range Section of a street or road normally identified with a number. For example, 1985 DOWNING LANE uses a number to denote where to deliver the letter or package on Downing Lane. If the address is other than conventional, the range field denotes the pertinent PO box or rural route number.

Region The first letter of the ZIP Code that indicates the region of the country.

Residential Delivery Indicator Residential Delivery Indicator (RDI) indicates whether an address is a residential delivery address (not a business delivery addresses).

Return Values A code that a program or subroutine issues to indicate the status of the processing performed. For example, the subroutine passes a return code to the calling (driver) program to indicate whether to assign a carrier route code to a given address.

SCF Sectional Center Facility, a major USPS mail collection and distribution center. For multi-ZIP cities, the first three digits of the ZIP Code the city indicate the SCF.

Sector Segment The ZIP add-on, or, commonly +4 code. See also ZIP Sector Number and ZIP Segment Number.

Street Direction Refers to the geographical location of any given street address (for example, North, South, East, or West).

Suffix Normally, a word that follows the street name, indicating the type of street. The following are common suffixes to the street name in a conventional address: Boulevard, Road, Lane, Avenue, Highway, Court, Street, and Drive.

Suite^{Link} The USPS Suite^{Link} database contains data on business addresses that were identified as high-rise default records during CASS processing. Finalist uses the USPS Suite^{Link} database to append the secondary (suite) information to business addresses identified in the input file as high-rise default records. Records that have been processed through CASS Certified™ ZIP + 4® matching software and identified as high-rise defaults are potential candidates for Suite^{Link} processing.

Three-Digit ZIP Prefix The first three digits of the ZIP Code. These digits determine the appropriate SCF or postal facility to which the mail should be routed.

Unassigned Address An address that does not have a sector segment number assigned by the post office.

Undefined Records Records of varying length that do not contain a record length field. An undefined record is equivalent to a block.

Unique ZIP Code A ZIP Code that is unique to a building or business.

Unit designator Type of unit (i.e., APT, STE, #, etc.)

URB See Urbanization.

Urbanization Denotes a sector, area, or development within a geographic area. Only used in Puerto Rico urban areas.

ZIP Code The Zoning Improvement Plan (ZIP), established in 1963, is a system of five-digit codes identifying the individual post office or metropolitan area delivery station associated with an address. ZIP Code is a USPS trademark.

ZIP Sector Number The ZIP sector number forms the first two digits of the ZIP add-on code. Geographically, a ZIP sector is a subdivision of a five-digit ZIP Code area. ZIP sector boundaries do not Finalist state or county lines.

ZIP Segment Number

The ZIP segment number forms the last two digits of the ZIP add-on code. The ZIP segment is a sub-division of a ZIP sector. Geographically, ZIP segments represent areas such as one side of a city block between intersections; both sides of a street, including cul-de-sacs; a company or building; a floor or group of floors within a building; a cluster of mailboxes; sections of post office boxes; or other similar delivery groups.

ZIP + 4 code

The nine-digit code, established in 1981 is composed of:

Initial Code — the first five digits identifying the post office or metropolitan area delivery station associated with an address; a hyphen.

Expanded Code — Includes the additional four digits. The first two additional digits designate the sector (a geographic portion of a zone, a portion of rural route, several city blocks or a large building, part of a box section, or an official designation). The last two digits designate the segment (a specific block face, apartment house bank of boxes, a firm, a floor in a large building, or other specific location). ZIP + 4 is a USPS trademark.

Zone — Last two digits of the ZIP Code; also, an area defined by the Postal Service for the purpose of establishing mailing rates. Mileage from a central mailing point determines all zones.

INDEX

G

I

K

L

P

R

S

U

Feedback

Document Title: Getting Started With Finalist Guide

Document Date: November 2012

Software Version: 8.2.0

Comment regarding page: 1

Contact Preferences

May we contact you if we have questions about your comments?

- ○ Yes
- ◉ No

Contact Information

Name:

Email:

Phone:

Best time to contact: Any time

Comments

Enter you comments here:

Send

Pitney Bowes Software
One Global View
Troy, New York 12180
www.pbinsight.com

Main: +1 (301) 731-2300
Sales: +1 (888) 413-6763
Support: +1 (800) 367-6950