

Address Matching System

Application Program Interface

User Guide  
April 2007

 **UNITED STATES  
POSTAL SERVICE.**

## Table of Contents

<b>Section 1: Introduction and Overview</b> .....	<b>2</b>
USPS Address Matching System Developer’s Kit .....	2
AMS CD-ROM Technical Support .....	2
Installation Procedures for Windows 32-Bit.....	3
Installation Procedures for SUN UNIX (32 Bit).....	5
Installation Procedures for SUN UNIX (64 Bit).....	7
Installation Procedures for AIX UNIX .....	9
Installation Procedures for LINUX.....	11
<b>Section 2: API Functions</b> .....	<b>13</b>
Open the Address Matching System – [Deprecated].....	14
Open the Address Matching System with Special Parameters .....	15
Address Inquiry.....	18
Address Sort Key .....	23
9-digit Inquiry .....	25
11-digit Inquiry.....	27
Address Standardization .....	29
Close the Address Matching System .....	31
Read City/State File By Key.....	32
Read City/State File Next .....	33
Read ZIP+4 File By Key.....	35
Read ZIP+4 File Next .....	36
Get ZIP Codes from a City/State .....	38
Terminate Active Address Inquiry.....	40
Get Date of ZIP+4 Database .....	41
Get AMS Data Expiration.....	43
Get AMS Library Expiration .....	45
Get CD-ROM Expiration Information – [Deprecated].....	47
Get API Code Version .....	49
Multiple Response Stack.....	50
Get Last Error.....	52
Get Environment.....	54
Retrieving the LACSLink Security Key.....	55
Checking for LACSLink Functionality.....	58
Disabling the LACSLink Security Key.....	60
<b>Section 3: Footnote Flags</b> .....	<b>62</b>
<b>Section 4: Record Types</b> .....	<b>65</b>
<b>Appendix A: Interface Definition</b> .....	<b>67</b>
<b>Appendix B: GDEV Application</b> .....	<b>76</b>

## **Section 1: Introduction and Overview**

The USPS *Address Matching System Application Programming Interface User Guide* is the primary reference document for the USPS National Customer Support Center's Address Matching System product. The guide contains installation instructions for each platform as well as function descriptions.

The USPS Address Matching System (AMS) is an application programming interface (API). As such, this guide should be used when the user wants to interface an application with the Address Matching System.

## **USPS Address Matching System Developer's Kit**

The USPS Address Matching System Developer's Kit contains the following:

- API library(s) for each specific computer platform
- Interface definition file (ZIP4.H)
- Test utility (SAMPLE.EXE)
- Test utility source code
- Sample configuration data files
- User documentation

The test utility can be used to ensure that the Address Matching System and data files have been installed correctly and to provide access to our matching logic, which displays the standardized address returned by the matching engine. This enables you to verify the accuracy of the ZIP+4 results returned from your product.

The AMS software, including, but not limited to, .DLLs, shared objects and static objects all expire and cease functionality based on USPS Coding Accuracy Support System (CASS) guidelines. The AMS software expires July 31st each year. The AMS data expires 105 days from the release date of the CDROM, which is the 15th day of each month.

During your development cycle and subsequent updates of your software, you should compile your software with the AMS library. The AMS library will handle any necessary interface with the DPV library and the KeyManager library.

## **AMS CD-ROM Technical Support**

If there are any questions regarding the Address Matching System API, please call the USPS' National Customer Support Center, AMS CD-ROM Technical Support at 1-877-640-0724. Hours of operation are 7am to 5pm Monday through Friday CT.



4. Use SAMPLE.C as an example to create your own API application.
5. Refer to Section 2, API Functions, to test other API function commands.
6. The following is an explanation of the API files for W32:

a.	ZIP4_W32.DLL	ZIP4 dynamic-link library
b.	ZIP4_W32.LIB	Stub library to link with the user application
c.	ZIP4.H	Interface header file
d.	Z4CONFIG.DAT	File location file [Deprecated]
e.	Z4CXLOG.DAT	Date time file
f.	SAMPLE.C	Sample C source file
g.	SAMPLE.H	Sample header file
h.	SAMPLE.EXE	Sample executable
i.	KEYMGR3.DLL	Key manager dll

### **Special Notes for Windows 32-bit**

The Windows 32-bit version of the Address Matching System DLL was built with all export functions having the ‘\_cdecl’ calling convention, which has caused problems with some programming languages that do not support this convention. To provide access to the address matching routines in the DLL for non C and C++ languages, the DLL also contains a set of routines with the proper DLL calling convention ‘\_stdcall.’ These routines have separate names from the original routines to preserve linkage with existing programs, and the new names are a concatenation of the original function name and ‘STD,’ which implies the \_stdcall calling convention, e.g.,

<b>_cdecl function name</b>	<b>_stdcall function name</b>
z4open()	z4openSTD()
z4adring()	z4adringSTD()
z4close()	z4closeSTD()

All of the \_stdcall functions map directly to the original functions, so there is no loss in functionality. All existing functions have an associated \_stdcall version, and all future additions to the DLL will contain both a \_cdecl version and a \_stdcall version.

## Installation Procedures for SUN UNIX (32 Bit)

1. Create a directory on your hard drive in which to store the API files.

Ex: `mkdir /usr/src/ams`

2. Copy the Address Matching System (AMS) files to your hard drive.

The AMS product is distributed on CD or DVD. The AMS files are located in the corresponding directory below:

[CD]/dev\_kits/sun/

[DVD]/<PRODUCT TYPE>/dev\_kits/sun/

All of the files in this directory are encrypted and must be unencrypted before use.

There are two (2) utility programs on the CD/DVD that will unencrypt the files.

- a. GDEV – See Appendix B for description and use.
- b. `dev_sun.exe` is located in the `dev_kits` directory

Ex: `DEV_SUN.EXE CUST_ID OUTPUT_PATH PRODUCT_FILE`

- a. `OUTPUT_PATH` is the directory created in step 1.
- b. `PRODUCT_FILE` is a file from the list in step 6. This should not include any directory paths.

The installation program must be executed from within the CD-ROM directory. This step needs to be performed once for each file listed in the file description in step 6 on the next page.

**Note:** *A customer ID (CUST\_ID) should be obtained from AMS CD-ROM Technical Support. The customer ID must be entered in uppercase letters.*

*The customer ID provided by AMS CD-ROM Technical Support will change each month. We do not recommend hard-coding the customer ID into an install program. For program installation, you may obtain a unique customer ID from AMS CD-ROM Technical Support. This unique customer ID will not change for the duration of the AMS API license unless otherwise specified.*

3. Run `SAMPLESH` and `SAMPLEST` to test AMS.
  - a. `chmod` on `samplesh` and `samplest` to `rxw`
  - b. `chmod` on `z4cxlog.dat` to `rw`
  - c. Select the option to manually enter the paths
4. Use `SAMPLE.C` as an example to create your own API application.
5. Refer to Section 2, API Functions, to test other API function calls.
6. The following is an explanation of the API files for SUN UNIX:
  - a. `LIBZ4SUN.SO`      ZIP4 shared library
  - b. `ZIP4_SUN.A`      Static link library; not recommended
  - c. `ZIP4.H`          Interface header file

- d. Z4CONFIG.DAT File location file [Deprecated]
- e. Z4CXLOG.DAT Date time file
- f. SAMPLE.C Sample C source file
- g. SAMPLE.H Sample header file
- h. SAMPLESH Sample executable linked with LIBZ4SUN.SO
- i. SAMPLEST Sample executable built with ZIP4\_SUN.A
- j. LIBKEYMGR.SO.3 Key manager shared library

### **Special Notes for SUN UNIX (32 Bit)**

The Address Matching System CD-ROM uses the ISO9660 file-system format, which stores file names in uppercase letters with a version control number appended to the end. The API requires that the CD-ROM file names appear in lowercase letters without the version number. Some versions of UNIX will automatically accommodate file name conversion during the mount process, but some require the user to specify the conversion explicitly with the options of the “mount” command. Please see the **man** pages on mount for more information on these options.

The Address Matching System SUN API Developer’s Kit contains both a static-link and a shared library. The static-link library is provided for compatibility with older programs written before the shared library was available. The USPS does not recommend use of the static-link library because logic changes are often made to the API, and the user would have to re-link the executable file with the AMS static-link library every time there is an update. Also, in compliance with CASS rules, the API code is set to expire at the end of the current CASS cycle, each August. If this date is reached without re-linking with a newer API, a user’s application will stop functioning.

To avoid these problems the USPS recommends using the AMS shared library so that user applications can gain immediate access to any logic changes simply by installing the new shared library. Unless otherwise specified, user applications do not need to be re-linked when a new shared library is provided on CD-ROM updates.

## Installation Procedures for SUN UNIX (64 Bit)

1. Create a directory on your hard drive in which to store the API files.  
Ex. `mkdir /usr/src/ams`
2. Copy the Address Matching System (AMS) files to your hard drive..

The AMS product is distributed on CD or DVD. The AMS files are located in the corresponding directory below:

[CD]/dev\_kits/s64/

[DVD]/<PRODUCT TYPE>/dev\_kits/s64/

All of the files in this directory are encrypted and must be unencrypted before use.

There are two (2) utility programs on the CD/DVD that will unencrypt the files.

- a. GDEV – See Appendix B for description and use
- b. `dev_s64.exe` located in the `dev_kits` directory

Ex. `DEV_S64.EXE CUST_ID OUTPUT_PATH PRODUCT_FILE`

- a. `OUTPUT_PATH` is the directory created in step 1.
- b. `PRODUCT_FILE` is a file from the list in step 6. This should not include any directory paths.

The installation program must be executed from within the CD-ROM directory. This step needs to be performed once for each file listed in the file description in step 6 on the next page.

**Note:** *A customer ID (CUST\_ID) should be obtained from AMS CD-ROM Technical Support. The customer ID must be entered in uppercase letters.*

*The customer ID provided by AMS CD-ROM Technical Support will change each month. We do not recommend hard-coding the customer ID into an install program. For program installation, you may obtain a unique customer ID from AMS CD-ROM Technical Support. This unique customer ID will not change for the duration of the AMS API license unless otherwise specified.*

3. Run SAMPLESH to test AMS.
  - a. `chmod` on `samplesh` to `rxw`
  - b. `chmod` on `z4cxlog.dat` to `rw`
  - c. Select the option to manually enter the paths
4. Use `SAMPLE.C` as an example to create your own API application.
5. Refer to Section 2, API Functions, to test other API function calls.
6. The following is an explanation of the API files for SUN UNIX:
  - a. `LIBZ4SUN64.SO` ZIP4 shared library
  - b. `ZIP4.H` Interface header file
  - c. `Z4CONFIG.DAT` File location file [Deprecated]
  - d. `Z4CXLOG.DAT` Date time file
  - e. `SAMPLE.C` Sample C source file

- f. SAMPLE.H            Sample header file
- g. SAMPLESH           Sample executable linked with LIBZ4SUN.SO
- h. LIBKEYMGR.SO.3    Key manager shared library

**Special Notes for SUN UNIX (64 Bit)**

The Address Matching System CD-ROM uses the ISO9660 file-system format, which stores file names in uppercase letters with a version control number appended to the end. The API requires that the CD-ROM file names appear in lowercase letters without the version number. Some versions of UNIX will automatically accommodate file name conversion during the mount process, but some require the user to specify the conversion explicitly with the options of the “mount” command. Please see the **man** pages on mount for more information on these options.

The Address Matching System S64 API Developer’s Kit contains a shared library. In compliance with CASS rules, the API code is set to expire at the end of the current CASS cycle, each August. If this date is reached without replacing the shared library, a user’s application will stop functioning.

## Installation Procedures for AIX UNIX

- 1 Create a directory on your hard drive in which to store the API files.

Ex. `mkdir /usr/src/ams`

- 2 Copy the Address Matching Engine (AMS) files to your hard drive.

The AMS product is distributed on CD or DVD. The AMS files are located in the corresponding directory below.

[CD]/dev\_kits/aix/

[DVD]/<PRODUCT TYPE>/dev\_kits/aix/

All of the files in this directory are encrypted and must be unencrypted before use.

There are two (2) utility programs on the CD/DVD that will unencrypt the files.

- a. GDEV – See Appendix B for description and use.
- b. `dev_aix.exe` located in the `dev_kits` directory

Ex. `DEV_AIX.EXE CUST_ID OUTPUT_PATH PRODUCT_FILE`

- a. `OUTPUT_PATH` is the directory created in step 1.
- b. `PRODUCT_FILE` is a file from the list in step 6. This should not include any directory paths

The installation program must be executed from within the CD-ROM directory. This step needs to be performed once for each file listed in the file description in step 6 on the next page.

**Note:** *A customer ID (CUST\_ID) should be obtained from AMS CD-ROM Technical Support. The customer ID must be entered in uppercase letters.*

*The customer ID provided by AMS CD-ROM Technical Support will change each month. We do not recommend hard-coding the customer ID into an install program. For program installation, you may obtain a unique customer ID from AMS CD-ROM Technical Support. This unique customer ID will not change for the duration of the AMS API license unless otherwise specified.*

- 3 Run `SAMPLEST` to test AMS

- a. `chmod` on `samplest` to `rxw`
- b. `chmod` on `z4cxlog.dat` to `rw`
- c. Select the option to manually enter the paths.

- 4 Use `SAMPLE.C` as an example to create your own API application.

- 5 Refer to Section 2, API Functions, to test other API function commands.

- 6 The following is an explanation of the API files for AIX UNIX:

- a. `ZIP4_AIX.A`           Static-link library
- b. `ZIP4.H`                Interface header file
- c. `Z4CONFIG.DAT`       File location file [Deprecated]
- d. `Z4CXLOG.DAT`        Date time file

- e. SAMPLE.C            Sample C source file
- f. SAMPLE.H            Sample header file
- g. SAMPLEST            Sample executable built with ZIP4\_AIX.A

### **Special Notes for AIX UNIX**

The Address Matching System CD-ROM uses the ISO9660 file-system format, which stores file names in uppercase letters with a version control number appended to the end. However, the API requires that the CD-ROM file names appear in lowercase letters without the version number. Some versions of UNIX will automatically accommodate file-name conversion during the mount process, but some require the user to specify the conversion explicitly with the options of the “mount” command. Please see the **man** pages on mount for more information on these options.

## Installation Procedures for LINUX

1. Create a directory on your hard drive in which to store the API files.

Ex. `mkdir /usr/src/ams`

2. Copy the Address Matching System (AMS) files to your hard drive.

The AMS product is distributed on CD or DVD. The AMS files are located in the corresponding directory below:

[CD]/dev\_kits/lnx/

[DVD]/<PRODUCT TYPE>/dev\_kits/lnx/

All of the files in this directory are encrypted and must be unencrypted before use.

There are two (2) utility programs on the CD/DVD that will unencrypt the files.

- a. GDEV – See Appendix B for description and use
- b. `dev_lnx.exe` located in the `dev_kits` directory

`DEV_LNX.EXE` located on the CD-ROM in the `DEV_KITS` directory.

Ex. `DEV_LNX.EXE CUST_ID OUTPUT_PATH PRODUCT_FILE`

- a. `OUTPUT_PATH` is the directory created in step 1.
- b. `PRODUCT_FILE` is a file from the list in step 6. This should not include any directory paths.

The installation program must be executed from within the CD-ROM directory. This step needs to be performed once for each file listed in the file description in step 6 on the next page. Following initial installation, the only files that need to be installed with subsequent CD-ROM updates are the header files and libraries. A batch file is recommended to simplify this install process.

**Note:** *A customer ID (CUST\_ID) should be obtained from AMS CD-ROM Technical Support. The customer ID must be entered in uppercase letters.*

*The customer ID provided by AMS CD-ROM Technical Support will change each month. We do not recommend hard-coding the customer ID into an install program. For program installation, you may obtain a unique customer ID from AMS CD-ROM Technical Support. This unique customer ID will not change for the duration of the AMS API license unless otherwise specified.*

3. Run `SAMPLESH` and `SAMPLEST` to test AMS.
  - a. `chmod` on `samplesh` and `samplest` to `rwx`
  - b. `chmod` on `z4cxlog.dat` to `rw`
  - c. Select the option to manually enter the paths
4. Use `SAMPLE.C` as an example to create your own API application.
5. Refer to Section 2, API Functions, to test other API function commands.
6. The following is an explanation of the API files for LNX:
  - c. `LIBZ4LNX.SO` ZIP4 shared library

- d. ZIP4\_LNX.A           Static link library; not recommended
- e. ZIP4.H                Interface header file
- f. Z4CONFIG.DAT        File location file [Deprecated]
- g. Z4CXLOG.DAT         Date time file
- h. SAMPLE.C             Sample C source file
- i. SAMPLE.H             Sample header file
- j. SAMPLESH            Sample executable linked with LIBZ4LNX.SO
- k. SAMPLEST            Sample executable built with ZIP4\_LNX.A
- l. LIBKEYMGR.SO.3      Key manager shared library

### **Special Notes for LINUX**

The Address Matching System CD-ROM uses the ISO9660 file -system format, which stores file names in uppercase letters with a version control number appended to the end. However, the API requires that the CD-ROM file names appear in lowercase letters without the version number. Some versions of UNIX will automatically accommodate file -name conversion during the mount process, but some require the user to specify the conversion explicitly with the options of the “mount” command. Please see the **man** pages on mount for more information on these options.

The Address Matching System LINUX API Developer’s Kit contains both a static-link and a shared library. The static-link library is provided for compatibility with older programs written before the shared library was available. The USPS does not recommend use of the static-link library because logic changes are often made to the API, and the user would have to re-link the executable files with the AMS staticlink library every time there is an update. Also, in compliance with CASS rules, the API code is set to expire at the end of the current CASS cycle, each August. If this date is reached without re-linking with a newer API, a user’s application will stop functioning.

To avoid these problems , the USPS recommends using the AMS shared library so that user applications can gain immediate access to any logic changes simply by installing the new shared library. Unless otherwise specified, user applications do not need to be re-linked when a new shared library is provided on CD-ROM updates.

**Section 2: API Functions**

- `z4open()` Open the Address Matching System
- `z4opencfg()` Open the Address Matching System with Special Parameters
- `z4adrinq()` Address Inquiry
- `z4adrkey()` Address Sort Key
- `z4xrfinq()` 9-digit Inquiry
- `z4xrfinq11()` 11-digit Inquiry
- `z4adrstd()` Address Standardization
- `z4close()` Close the Address Matching System
- `z4ctyget()` Read City/State File by Key
- `z4ctynxt()` Read City/State File Next
- `z4adrget()` Read ZIP+4 File by Key
- `z4adrnxt()` Read ZIP+4 File Next
- `z4getzip()` Get a ZIP Code range for a City/St
- `z4abort()` Terminate Active Address Inquiry
- `z4date()` Get Date of ZIP+4 Database
- `z4GetDataExpireDays()` Get AMS Data Expiration
- `z4GetCodeExpireDays()` Get AMS Library Expiration
- `z4expire()` Get CD ROM Expiration Information- Deprecated: Use `z4GetDataExpireDays()`
- `z4ver()` Get the Version of the API code
- `z4scroll()` Multiple Response Stack
- `z4geterror()` Get Last Error
- `z4getenv()` Get Environment
- `z4LLkGetKey()` Retrieving the LACSLink Security Key
- `z4LLkIsDisabled()` Checking for LACSLink Functionality
- `z4LLkSetKey()` Disabling the LACSLink Security Key

## Open the Address Matching System – [Deprecated]

The `z4open()` function opens the Address Matching System for application use. This function must be called before attempting to use any of the inquiry functions. During system opening, the Address Matching System allocates memory buffers and file handles for disk I/O. The function returns a code summarizing the results of the open operation.

It is recommended that you use the `z4opencfg()` function (see page 15) instead of the `z4open()` function. The `z4open()` function searches the systems for a configuration file and will use the first one found. This function can cause unexpected operation by using a configuration file that was not expected to be in its search path.

**Note:** *The `z4open()` function does not provide access to eLOT, DPV or LACSLink. Beginning with the CASS 2007-2008 (Cycle L) release, these components are required and, as a result, all uses of this function must be changed to use `z4opencfg()` (see page 15).*

### Syntax

```
#include <zip4.h>
int z4open(void);
```

### Input

None

### Output

None

## Open the Address Matching System with Special Parameters

The `z4opencfg()` function opens the Address Matching System in the same manner as `z4open()`, but it gives the user more control over the configuration file and Enhanced Line of Travel (eLOT) processing. Enhanced Line of Travel is available through the USPS AMS API, but it is turned off by default. To enable eLOT processing, you must first call `z4opencfg()` and set the `elotflag` variable to 'Y'. You must also use the `CONFIG_PARM` to specify the paths to the AMS database.

**Note:** *The DPV and LACS<sup>Link</sup>™ components are no longer optional and must always be enabled. While the Z4OPEN\_PARM still contains the `llkflag` and `dpvflag` variables, they no longer provide any functionality and are ignored by the AMS library.*

### Syntax

```
#include <zip4.h>
int z4opencfg(Z4OPEN_PARM* openparm);
```

### Input

`openparm`            pointer to a Z4OPEN\_PARM structure

If a field in the Z4OPEN\_PARM is not used, then it must be initialized to zero (see example code).

```
typedef struct
{
    char    rsvd1[50];

    short   status;
    char    *fname

    CONFIG_PARM config

    char    elotflag;
    char    llkflag;
    char    dpvflag;
    char    systemflag;

    char    rsvd2[512];
}Z4OPEN_PARM;
```

### Field Definitions:

<code>rsvd1</code>	Reserved for future use.
<code>status</code>	See "Output" section.
<code>fname</code>	Pointer to a string that contains the full path and filename for a custom config file. [Deprecated]
<code>config</code>	Embedded structure for setting the path names to the AMS database. (Not used if <code>fname</code> is set)
<code>elotflag</code>	Set to 'Y' to activate eLOT processing.
<code>llkflag</code>	Usage has been discontinued
<code>dpvflag</code>	Usage has been discontinued
<code>systemflag</code>	Set to 'Y' to de-activate the auto-generation of the security file.
<code>rsvd2</code>	Reserved for future use.

## Output

Z4OPEN\_PARM.status will be set to 1, 2 or 9 to indicate which value was used for the configuration file.

<u>Name</u>	<u>Value</u>	<u>Meaning</u>
Z4_FNAME	1	Used the value pointed to by the <i>fname</i> character pointer
Z4_CONFIG	2	Used the values pointed to by the CONFIG_PARM structure
Z4_SEARCH	9	Searched for a file named z4config.dat

## Return

- 1 The USPS Address Matching System is already open
- 0 The USPS Address Matching System opened successfully
- 1 The USPS Address Matching System is not in sync
- 2 The USPS Address Matching System has expired
- 4 The USPS Address Matching System failed to open DPV
- 5 The USPS Address Matching System failed to open DPV
- 7 The USPS Address Matching System failed to open LACSLink

**Note:** See the DPV User Guide for specific information on DPV errors.

## Example

```
#include <stdio.h>
#include <zip4.h>

void main(void)
{
    Z4OPEN_PARM openparm;
    int rtn=0;

    memset(&openparm, 0, sizeof(openparm));

    /*Open with the paths embedded in the CONFIG_PARM structure*/
    openparm.config.address1    = "c:\\amsdata\\";
    openparm.config.addrindex  = "c:\\amsdata\\";
    openparm.config.cdrom      = "d:\\";
    openparm.config.citystate  = "c:\\amsdata\\";
    openparm.config.crossref   = "c:\\amsdata\\";
    openparm.config.system     = "c:\\amsdata\\";
    openparm.config.elot       = "c:\\elotdata\\";
    openparm.config.elotindex  = "c:\\elotdta\\";
    openparm.config.llkpath    = "c:\\llkdata\\";
    openparm.config.dpvpath    = "c:\\dpvdata\\";
    openparm.config.fnsnpath   = "c:\\amsdata\\";

    /*Turn eLOT processing on*/
    openparm.elotflag = 'Y';

    rtn = z4opencfg(&openparm);

    if(rtn==0)
```

```
        printf("\nSuccess opening the USPS Address Matching System.");  
else  
        printf("\nError opening the USPS Address Matching System.");  
  
/*close the USPS Address Matching System*/  
z4close();  
}
```

## Address Inquiry

The `z4adrinq()` function commands the Address Matching System to perform an address inquiry using firm name (optional), address, and city/state/ZIP information. Before performing this function, the input address information must be copied into the corresponding input fields outlined below. Note that the City, State, and ZIP fields may be placed either within the `parm. ictyi` field or copied to the `parm. ictyi`, `parm. istai`, and `parm. izipc` fields, respectively. Following the address inquiry, the `parm. retcc` field contains a response code summarizing the inquiry results. If an address response was found, standardized address information will be located in the output fields described below.

### Syntax

```
#include <zip4.h>

int z4adrinq(ZIP4_PARM *parm);
```

### Input

The `parm` argument must point to a `ZIP4_PARM` structure. The following fields must be initialized before calling the `z4adrinq()` function. If a field is not used, it must be initialized to zero.

<code>parm.iadl1</code>	Street Address
<code>parm.iadl2</code>	Firm Name
<code>parm.iadl3</code>	Secondary Address
<code>parm.iprurb</code>	Puerto Rican Urbanization Name
<code>parm.ictyi</code>	City or City/State/ZIP
<code>parm.istai</code>	State or empty
<code>parm.izipc</code>	ZIP or empty

### Output

<b>parm.retcc</b>	<b>Response code</b>
<code>Z4_SINGLE</code>	31 — A single address was found
<code>Z4_DEFAULT</code>	32 — An address was found, but a more specific address could be found with more information
<b>parm.retcc</b>	<b>Response Code</b>
<code>Z4_INVADDR</code>	10 — Invalid input address (i.e., contained a dual address)
<code>Z4_INVZIP</code>	11 — Invalid input 5-digit ZIP Code
<code>Z4_INVSTATE</code>	12 — Invalid input state abbreviation code
<code>Z4_INVCITY</code>	13 — Invalid input city name
<code>Z4_NOTFND</code>	21 — No match found using input address
<code>Z4_MULTIPLE</code>	22 — Multiple responses were found and more specific information is required to select a single or default response

**parm.foot****Footnotes**

parm.foot.a = "A"	ZIP Code Corrected
parm.foot.b = "B"	City/State Corrected
parm.foot.c = "C"	Invalid City/State/ZIP
parm.foot.d = "D"	No ZIP+4 Code Assigned
parm.foot.e = "E"	ZIP Code Assigned with a Multiple Response
parm.foot.f = "F"	Address Not Found
parm.foot.g = "G"	All or Part of the Firm Line Used For Address Line
parm.foot.h = "H"	Missing Secondary Number
parm.foot.i = "I"	Insufficient/Incorrect Data
parm.foot.j = "J"	Dual Address on Input
parm.foot.k = "K"	Multiple response due to Cardinal Rule
parm.foot.l = "L"	Address Component Changed
parm.foot.m = "M"	Street Name Changed
parm.foot.n = "N"	Address Standardized
parm.foot.p = "P"	Better Address Exists
parm.foot.q = "Q"	Unique ZIP Code Match
parm.foot.r = "R"	No Match due to EWS
parm.foot.s = "S"	Incorrect Secondary Number
parm.foot.t = "T"	Multiple response due to Magnet Street Syndrome
parm.foot.u = "U"	Unofficial Post Office Name
parm.foot.v = "V"	Unverifiable City/State
parm.foot.w = "W"	Small Town Default
parm.foot.x = "X"	Unique ZIP Code Generated
parm.foot.y = "Y"	Military Match
parm.foot.z = "Z"	ZIP Move Match

**Return Address Description**

parm.dadl1	Standardized Output Address
parm.dadl2	Standardized Output Firm Name
parm.dadl3	Standardized Secondary Address
parm.dprurb	Standardized Puerto Rican Urbanization Name
parm.dctya	Standardized Output City
parm.dstaa	Standardized Output State
parm.zipc	5-digit ZIP Code
parm.addon	4-digit Add-on Code
parm.cris	4-digit Carrier Route Code
parm.county	3-digit County Code

parm.dpbc	2-digit Delivery Point Barcode and 1-digit Check Digit
parm.mpnum	Matched Primary Number
parm.msnum	Matched Secondary Number
parm.auto_zone_ind	Carrier Route Rate Sort Indicator (Y or N)
parm.elot_num	Enhanced Line of Travel (eLOT) number
parm.elot_code	eLOT Ascending/Descending Flag (A/D)
parm.llk_rc	LACSLink Return Code
parm.llk_ind	LACSLink Indicator

<b>Parsed Input</b>	<b>Description</b>
ppnum	Primary Number
psnum	Secondary Number
prote	Rural Route Number
punit	Secondary Number Unit
ppre1	First or Left Pre-direction
ppre2	Second or Right Pre-direction
psuf1	First or Left Suffix
psuf2	Second or Right Suffix
ppst1	First or Left Post-direction
ppst2	Second or Right Post-direction
ppnam	Primary Name

### **Return**

- 0 - The USPS Address Matching System resident
- 1 - The USPS Address Matching System issued a system error
- 2 - The USPS Address Matching System not ready
- 3 - CD-ROM has expired

### **Additional Information About Z4ADRINQ()**

If parm.retcc is Z4\_INVADDR, Z4\_INVZIP, Z4\_INVSTATE, Z4\_INVCITY, Z4\_NOTFND, or Z4\_MULTIPLE, then the return address fields will contain the input address. If the input address is unambiguously a rural route, highway contract, PO box, or general delivery address, then the return fields will contain the normalized version of the input address.

If parm.retcc is Z4\_MULTIPLE, then parm.foot, parm.respn, and parm.stack are also returned by the system. The parm.zipc and/or parm.cris fields may contain data if all records in the stack have the same ZIP Code and/or carrier route ID.

If parm.retcc is Z4\_SINGLE or Z4\_DEFAULT, then all fields in the returned data section are returned by the Address Matching System. The first record in the parm.stack structure will contain the ZIP+4 record

to which the system matched. This record may be used to access the individual fields from the matched record, such as primary name, suffix, post-directional, etc.

### Example

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <zip4.h>

ZIP4_PARM parm;

int main(int argc, char** argv)
{
    Z4OPEN_PARM openparm;

    memset (&openparm, 0, sizeof(openparm));

    /* ... Populate openparm ... */

    /* open the USPS Address Matching System */
    if (z4opencfg(&openparm) != 0)
    {
        printf("The USPS Address Matching System failed to open");

        /* Always call z4close() even on open failure */
        z4close();

        exit(5);
    }

    /* load input address parameters */
    memset(&parm, 0, sizeof(parm));

    strcpy(parm.iadl2, "ACME TOOL AND DIE"); /* Firm line */
    strcpy(parm.iadl3, "STE 200" ); /* Secondary or extra line */
    strcpy(parm.iadl1, "323 S 152ND ST" ); /* Primary address line */
    strcpy(parm.iprurb, "" ); /* Puerto Rico specific */
    strcpy(parm.ictyi, "OMAHA, NE 68154" ); /* City, State, ZIP */

    /* request address inquiry */
    z4adrinq(&parm);

    /* if a response found (either single or default) */
    if(parm.retcc==Z4_SINGLE || parm.retcc==Z4_DEFAULT)
    {
        printf("Found response.\n");
        printf("Name: %s\n", parm.dadl2);
        printf("S Addr: %s\n", parm.dadl3);
        printf("Addr: %s\n", parm.dadl1);
        printf("PRUrb: %s\n", parm.dprurb);
        printf("City: %s\n", parm.dctya);
        printf("ST: %s\n", parm.dstaa);
        printf("ZIP: %s\n", parm.zipc);
        printf("Addon: %s\n", parm.addon);
        printf("DPBC: %s\n", parm.dpbc);
        printf("Pre Dir: %s\n", parm.stack[0].pre_dir);
    }
}
```

```
        printf("Str Name:      %s\n", parm.stack[0].str_name);
        printf("Suffix:      %s\n", parm.stack[0].suffix);
        printf("Post Dir:    %s\n", parm.stack[0].post_dir);
        printf("Lacs Ind:    %c\n", parm.stack[0].lacs_status);
    }

    /* close The USPS Address Matching System */
    z4close();

    exit(0)
}
```

## Address Sort Key

The `z4adrkey()` function creates a sort key for an address. This function can be used in batch processes to sort an input file in the order that addresses are contained on the Address Matching System data files. However, the function does not sort your file; it produces a key field to assist your software in sortation. Sorting an input file usually produces a dramatic increase in processing throughput.

### Syntax

```
#include <zip4.h>

int z4adrkey(ZIP4_PARM *parm);
```

### Input

The `parm` argument must point to a `ZIP4_PARM` structure. The following fields must be initialized before calling the `z4adrkey()` function.

<code>parm.iadl1</code>	Street Address
<code>parm.iadl2</code>	Firm Name
<code>parm.iprurb</code>	Puerto Rican Urbanization Name
<code>parm.ictyi</code>	City or City/ State/ ZIP
<code>parm.istai</code>	State or empty
<code>parm.izipc</code>	ZIP or empty

### Output

<code>parm.adrkey</code>	Address Sort Key
--------------------------	------------------

**Note:** *The contents and length of the address sort key are subject to change at any time. The key contains binary data and should be used in its entirety for the sort process.*

### Return

- 0 - The USPS Address Matching System resident
- 1 - The USPS Address Matching System issued a system error
- 2 - The USPS Address Matching System not ready

### Example

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <zip4.h>

ZIP4_PARM parm;

int main(int argc, char** argv)
{
    int i;

    Z4OPEN_PARM openparm;

    memset (&openparm, 0, sizeof(openparm));

    /* ... Populate openparm ... */

    /* open the USPS Address Matching System */
    if (z4opencfg(&openparm) != 0)
    {
```

```

printf("The USPS Address Matching System failed to open");

/* Always call z4close() even on open failure */
z4close();

exit(5);
}

/* load input address parameters */
memset(&parm, 0, sizeof(parm));

strcpy(parm.iadl2, "ACME TOOL AND DIE"); /* Firm line */
strcpy(parm.iadl3, "STE 200" ); /* Secondary or extra line */
strcpy(parm.iadl1, "323 S 152ND ST" ); /* Primary address line */
strcpy(parm.iprurb, "") /* Puerto Rico specific */
strcpy(parm.ictyi, "OMAHA, NE 68154" ); /* City, State, ZIP */

/* request address sort key */
z4adrkey(&parm);

/* print the address sort key in hex */
for(i=0; i<sizeof(parm.adrkey); i++)
    printf("%02X", parm.adrkey[i]);

printf("\n");

/* close The USPS Address Matching System */
z4close();
exit(0);
}

```

## 9-digit Inquiry

The `z4xrfinq()` (9-digit Inquiry) function commands the Address Matching System to perform an address inquiry using an input 9-digit ZIP Code. Before using this function, the input 9-digit ZIP Code must be copied into the `parm.iad11` field outlined below. Following the 9-digit inquiry, the `parm.retcc` field displays a return code summarizing the result of the inquiry. If an address response was found, standardized address information can be found in the output fields described in the Address Inquiry function description (see page 18).

**Note:** *This function only returns matches to address records, not to specific addresses. Address records generally contain a range of possible addresses.*

*To find a match to a specific address using only the ZIP Code, you will need to use the 11-Digit Inquiry function (see page 27).*

### Syntax

```
#include <zip4.h>
int z4xrfinq(ZIP4_PARM *parm);
```

### Input

The `parm` argument must point to a `ZIP4_PARM` structure. The following field must be initialized before calling the `z4xrfinq()` function:

`parm.iad11` 9-digit ZIP Code.

**Note:** *Return Code 22 denotes multiple responses. The address fields contain the first of a stack of ten possible responses (or matches).*

### Output

<b>parm.retcc</b>	<b>Response code</b>
Z4_SINGLE	A single address was found
Z4_DEFAULT	A default address was found, but more specific addresses exist
Z4_NOTFND	No match found; considered a not found address
Z4_MULTIPLE	Multiple responses were found

Refer to the Address Inquiry function description for other output fields (see page 18).

### Return

- 0 - The USPS Address Matching System resident
- 1 - The USPS Address Matching System issued a system error
- 2 - The USPS Address Matching System not ready
- 3 - The USPS Address Matching System has expired

### Example

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <zip4.h>
```

```

ZIP4_PARM parm;

int main(int argc, char** argv)
{
    Z4OPEN_PARM openparm;

    memset (&openparm, 0, sizeof(openparm));

    /* ... Populate openparm ... */

    /* open the USPS Address Matching System */
    if (z4opencfg(&openparm) != 0)
    {
        printf("The USPS Address Matching System failed to open");

        /* Always call z4close() even on open failure */
        z4close();

        exit(5);
    }

    /* load input 9-digit ZIP parameter */
    memset(&parm, 0, sizeof(parm));
    strcpy(parm.iadl1, "681642815");

    /* request address inquiry */
    z4xrfinq(&parm);

    /* if a response found (either single or default) */
    if(parm.retcc == Z4_SINGLE || parm.retcc == Z4_DEFAULT)
    {
        printf("Found response.\n");
        printf("Name:      %s\n", parm.dadl2);
        printf("Addr:      %s\n", parm.dadl1);
        printf("PRUrb:    %s\n", parm.dprurb);
        printf("City:     %s\n", parm.dctya);
        printf("ST:      %s\n", parm.dstaa);
        printf("ZIP:     %s\n", parm.zipc);
        printf("Addon:   %s\n", parm.addon);
        printf("DPBC:   %s\n", parm.dpbc);
    }

    /* close The USPS Address Matching System */
    z4close();

    exit(0);
}

```

## 11-digit Inquiry

The `z4xrfinq11()` (11-digit Inquiry) function commands the Address Matching System to perform an address inquiry using an input 11-digit ZIP Code. Before using this function, the input 11-digit ZIP Code must be copied into the `parm.iad11` field outlined below. Following the 11-digit inquiry, the `parm.retcc` field displays a return code summarizing the result of the inquiry. If an address response was found, standardized address information can be found in the output fields described in the Address Inquiry function description (see page 18).

### Syntax

```
#include <zip4.h>
int z4xrfinq11(ZIP4_PARM *parm);
```

### Input

The `parm` argument must point to a `ZIP4_PARM` structure. The following field must be initialized before calling the `z4xrfinq11()` function:

`parm.iad11` 11-digit ZIP Code

**Note:** *Return Code 22 denotes multiple responses. The address fields contain the first of a stack of ten possible responses (or matches). It is recommended that the first address in the output fields not be used as a mailing address because it is not an exact match.*

### Output

`parm.retcc` Response code

`Z4_SINGLE` A single address was found

`Z4_DEFAULT` A default address was found, but more specific addresses exist

`Z4_NOTFND` No match found; considered a not found address

`Z4_MULTIPLE` Multiple responses were found

Refer to the Address Inquiry function description for other output fields (see page 18).

### Return

0 - The USPS Address Matching System resident

1 - The USPS Address Matching System issued a system error

2 - The USPS Address Matching System not ready

3 - The USPS Address Matching System has expired

### Example

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <zip4.h>
```

```
ZIP4_PARM parm;
```

```
int main(int argc, char** argv)
```

```

{
    Z4OPEN_PARM openparm;

    memset (&openparm, 0, sizeof(openparm));

    /* ... Populate openparm ... */

    /* open the USPS Address Matching System */
    if (z4opencfg(&openparm) != 0)
    {
        printf("The USPS Address Matching System failed to open");

        /* Always call z4close() even on open failure */
        z4close();

        exit(5);
    }

    /* load input 11-digit ZIP parameter */
    memset(&parm, 0, sizeof(parm));

    strcpy(parm.iadl1, "68164281527");

    /* request address inquiry */
    z4xrfinql1(&parm);

    /* if a response found (either single or default) */
    if(parm.retcc == Z4_SINGLE || parm.retcc == Z4_DEFAULT)
    {
        printf("Found response.\n");
        printf("Name:  %s\n", parm.dadl2);
        printf("Addr:  %s\n", parm.dadl1);
        printf("PRUrb: %s\n", parm.dprurb);
        printf("City:  %s\n", parm.dctya);
        printf("ST:    %s\n", parm.dstaa);
        printf("ZIP:   %s\n", parm.zipc);
        printf("Addon: %s\n", parm.addon);
        printf("DPBC:  %s\n", parm.dpbc);
    }

    /* close The USPS Address Matching System */
    z4close();

    exit(0);
}

```

## Address Standardization

The `z4adrstd()` (Address Standardization) function instructs the Address Matching System to standardize an address. This function can be used when a `Z4_MULTIPLE` response is returned from the `z4adrinq()` function. Use this function to standardize an address from the stack, but use it with caution. The index parameter is relative to zero and must be in increments of ten for each `z4scroll()` function called. Therefore, the index will have a value between zero and `parm.respn` minus one. Do not use the offset into the current stack of ten records.

When this function is called, the record corresponding to the index value is moved to the first position on the stack (offset zero). If components from the `ADDR_REC` structure are needed for the current record that was processed through `z4adrstd()`, they may be retrieved from the first stack record. Do not use the modulus 10 of the index (`index % 10`) to retrieve the `ADDR_REC` components from the stack.

**Note:** *This function should only be used when an operator is reviewing the multiple responses returned and selecting the record to be standardized. Please be advised that using this function in an unattended (batch) mode may result in inaccurate matches and possible failure to CASS certify.*

### Syntax

```
#include <zip4.h>
int z4adrstd(ZIP4_PARM *parm, int index)
```

### Input

<code>parm</code>	Unmodified parameter list from previous call to <code>z4adrinq()</code> .
<code>index</code>	Index of stack record to standardize address (refer to the description above). This must be less than <code>parm.respn</code> .

### Output

<code>parm.dadl1</code>	Standardized Street Address
<code>parm.dadl2</code>	Standardized Firm Name
<code>parm.dprurb</code>	Standardized Puerto Rican Urbanization Name
<code>parm.dlast</code>	Standardized City/State/ZIP

### Return

- 0 - Success
- 1 - Failure (i.e., invalid index parameter)
- 2 - The USPS Address Matching System not ready

### Example

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <zip4.h>
ZIP4_PARM parm;
```

```

int main(int argc, char** argv)
{
    Z4OPEN_PARM openparm;

    memset (&openparm, 0, sizeof(openparm));

    /* ... Populate openparm ... */

    /* open the USPS Address Matching System */
    if (z4opencfg(&openparm) != 0)
    {
        printf("The USPS Address Matching System failed to open");

        /* Always call z4close() even on open failure */
        z4close();

        exit(5);
    }

    /* load input address parameters */
    memset(&parm, 0, sizeof(parm));

    strcpy(parm.iadl2, "ACME TOOL AND DIE"); /* Firm line */
    strcpy(parm.iadl3, ""); /* Secondary or extra line */
    strcpy(parm.iadl1, "1336 CHATMAN"); /* Primary address line */
    strcpy(parm.iprurb, ""); /* Puerto Rico specific */
    strcpy(parm.ictyi, "CORDOVA TN 38018"); /* City, State, ZIP */

    /* request address inquiry */
    z4adrinq(&parm);

    /* standardize second address */
    z4adrstd(&parm, 1);

    /* display address */
    printf("Found response.\n");
    printf("Name: %s\n", parm.dadl2);
    printf("Addr: %s\n", parm.dadl1);
    printf("PRUrb: %s\n", parm.dprurb);
    printf("City: %s\n", parm.dctya);
    printf("ST: %s\n", parm.dstaa);
    printf("ZIP: %s\n", parm.zipc);
    printf("Addon: %s\n", parm.addon);
    printf("DPBC: %s\n", parm.dpbc);

    /* close The USPS Address Matching System */
    z4close();

    exit(0);
}

```

## Close the Address Matching System

The `z4close()` function closes the Address Matching System and is called when address inquiries have been completed and the interface is no longer needed. During execution of this function, memory buffers and file handles allocated during the `z4open()` function are de-allocated and closed.

**Note:** *The `z4close()` function call must be called after all calls to the `z4opencfg()` function call – regardless if `z4opencfg()` succeeded or failed.*

### Syntax

```
#include <zip4.h>
int z4close(void);
```

### Input

None

### Output

None

### Return

- 0 - The USPS Address Matching System closed
- 1 - The USPS Address Matching System not resident
- 2 - The USPS Address Matching System not ready

### Example

```
#include <stdio.h>
#include <zip4.h>

void main(void)
{
    /* close The USPS Address Matching System */
    if(z4close() == 0)
        printf("The USPS Address Matching System closed.\n");
    else
        printf("Error closing the USPS Address Matching System.\n");
}
```

## Read City/State File By Key

The `z4ctyget()` (Read City/State File By Key) function initiates a read of the City/State File. A specific ZIP Code can be selected as a starting point in a read of the City/State File. To read subsequent records, the Read City/State File Next function is used. For documentation on the City/State File, please refer to the *Address Information System Products Technical Guide*, which is available from the USPS National Customer Support Center's Customer Support Department at 800-238-3150. It is also available on the Internet at <http://ribbs.usps.gov/files/addressing/pubs>

### Syntax

```
#include <zip4.h>

int z4ctyget(CITY_REC *cityrec, char *zipcode);
```

### Input

The CITYREC argument must point to a CITY\_REC structure. The contents of the structure will be altered to contain the first city for the requested ZIP Code. The ZIP Code argument must point to a valid 5-digit ZIP Code or "00000"

### Output

None

### Return

- 0 - Success
- 1 - Failure
- 2 - The USPS Address Matching System not ready

### Example

See example code for "Read City/State File Next" (page 33).

## Read City/State File Next

The `z4ctynxt()` (Read City/State File Next) function reads subsequent records of the City/State File. It can only be used after the `z4ctyget()` function has been called.

**Note:** *Multiple calls to `z4ctynxt()` can not be mixed with calls to other Address Matching System functions. This function is designed to be called after a `z4ctyget()` or previous `z4ctynxt()` function call. The results of the `z4ctynxt()` are undefined if it is called after any other AMS function call.*

### Syntax

```
#include <zip4.h>
int z4ctynxt(CITY_REC *cityrec);
```

### Input

The `CITY_REC` argument must point to a `CITY_REC` structure. The contents of the structure will be altered to contain the next city.

### Output

None

### Return

- 0 - Success
- 1 - Failure
- 2 - The USPS Address Matching System not ready

### Example

```
#include <stdio.h>
#include <stdlib.h>
#include <zip4.h>

CITY_REC city;

int main(int argc, char** argv)
{
    int i;

    Z4OPEN_PARM openparm;

    memset (&openparm, 0, sizeof(openparm));

    /* ... Populate openparm ... */

    /* open the USPS Address Matching System */
    if (z4opencfg(&openparm) != 0)
    {
        printf("The USPS Address Matching System failed to open");

        /* Always call z4close() even on open failure */
        z4close();

        exit(5);
    }
}
```

```
/* read first city */
z4ctyget(&city, "00000");

/* read 10 more cities */
for(i=0; i<10 && z4ctynext(&city) == 0; ++i)
{
    printf("%s %-28.28s %s %s\n" city.zip_code,
           city.city_name,
           city.state_abbrev,
           city.finance);
}

/* close The USPS Address Matching System */
z4close();
exit(0);
}
```

## Read ZIP+4 File By Key

The `z4adrget()` (Read ZIP+4 File by Key) function is used to read the ZIP+4 File. For documentation on the ZIP+4 File, please refer to the *Address Information Products Technical Guide*, which is available from the USPS National Customer Support Center's Customer Support Department at 800-238-3150. It is also available on the Internet at <http://ribbs.usps.gov/files/addressing/pubs>

A specific postal finance number can be selected as a starting point in a read of the ZIP+4 File. To read subsequent records, the `z4adrnxt()` function is used.

### Syntax

```
#include <zip4.h>
int z4adrget(ADDR_REC *addrrec, char *finance);
```

### Input

The ADDRREC argument must point to an ADDR\_REC structure. The contents of the structure will be altered to contain the first address for the requested postal finance number. The finance argument must contain a valid postal finance number or "000000"

### Output

None

### Return

- 0 - Success
- 1 - Failure
- 2 - The USPS Address Matching System not ready

### Example

See example code for "Read ZIP+4 File Next" (page 36).

## Read ZIP+4 File Next

The `z4adrnxt()` (Read ZIP+4 File Next) function reads subsequent records of the ZIP+4 File. It can only be used after the `z4adrget()` function has been called.

**Note:** *Multiple calls to `z4adrnxt()` can not be mixed with calls to other Address Matching System functions. This function is designed to follow a `z4adrget()` or another `z4adrnxt()` function call. The results of `z4adrnxt()` are undefined if it is called after any other AMS function*

### Syntax

```
#include <zip4.h>
int z4adrnxt(ADDR_REC *addrrec);
```

### Input

The `ADDRREC` argument must point to a `ADDR_REC` structure. The contents of the structure will be altered to contain the next address.

### Output

None

### Return

- 0 - Success
- 1 - Failure
- 2 - The USPS Address Matching System not ready

### Example

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <zip4.h>

CITY_REC city;
ADDR_REC addr;

int main(int argc, char** argv)
{
    Z4OPEN_PARM openparm;

    memset (&openparm, 0, sizeof(openparm));

    /* ... Populate openparm ... */

    /* open the USPS Address Matching System */
    if (z4opencfg(&openparm) != 0)
    {
        printf("The USPS Address Matching System failed to open");

        /* Always call z4close() even on open failure */
        z4close();

        exit(5);
    }
}
```

```
/* read a city */
z4ctyget(&city, "00000");

/* read first address record for this city */
z4adrget(&addr, city.finance);

/* read remaining adrs for this finance number */
while(z4adrnxt(&addr) == 0)
{
    /* check if finance number has changed */
    if (memcmp(addr.finance, city.finance, 6) != 0)
        break;

    /* Code to process the current address record. */
}

/* close The USPS Address Matching System */
z4close();

exit(0);
}
```

## Get ZIP Codes from a City/State

The `z4getzip()` (Get ZIP Codes) from a City/State function retrieves a range of ZIP Codes for a city or state and returns the valid high and the low values for the input city/state. The standardized form of the input city/state as well as the finance number are also returned.

*Note: All ZIP Codes within the range are not necessarily valid.*

### Syntax

```
#include <zip4.h>

int z4getzip(GET_ZIPCODE_STRUCT *parm);
```

### Input

The *parm* argument must point to a `GET_ZIPCODE_STRUCT` structure. The contents of the structure will be altered to contain the ZIP Code range for the input city/state.

`parm.input_cityst`      Input city/state to lookup

### Output

`parm.output_cityst`      Standardized city/state

`parm.low_zipcode`      Low ZIP Code value

`parm.high_zipcode`      High ZIP Code value

`parm.finance_num`      Finance number

### Return

0 - Success

1 - Failure

### Example

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <zip4.h>

GET_ZIPCODE_STRUCT parm;

int main(int argc, char** argv)
{
    Z4OPEN_PARM openparm;

    int result;

    memset (&openparm, 0, sizeof(openparm));

    /* ... Populate openparm ... */

    /* open the USPS Address Matching System */
    if (z4opencfg(&openparm) != 0)
    {
        printf("The USPS Address Matching System failed to open");

        /* Always call z4close() even on open failure */
        z4close();
    }
}
```

```
        exit(5);
    }

    /* read a city */
    strcpy(parm.input_cityst, "MEMPHIS TN");
    result=z4getzip(&parm);

    /* Display the ZIP codes found */
    if(result == 0)
    {
        printf("CITY FOUND:      %s\n",parm.output_cityst);
        printf("LOW ZIP:         %s\n",parm.low_zipcode);
        printf("HIGH ZIP:          %s\n",parm.high_zipcode);
        printf("FINANCE:           %s\n",parm.finance_num);
    }

    /* close The USPS Address Matching System */
    z4close();

    exit(0);
}
```

## Terminate Active Address Inquiry

The `z4abort()` (Terminate Active Address Inquiry) function terminates an active address inquiry and is useful in real-time applications where each inquiry must be completed within a specified period of time. This function would normally be called from within a timer interrupt handler. The `z4adrinq()` call in progress is terminated by the function call.

### Syntax

```
#include <zip4.h>
int z4abort(void);
```

### Input

None

### Output

None

### Return

None

## Get Date of ZIP+4 Database

The `z4date()` (Get Date of ZIP+4 Database) function returns the date of the ZIP+4 database and prints the date for PS Form 3553 (CASS certificate). The date is returned as an 8-byte character string in the “YYYYMMDD” format.

### Syntax

```
#include <zip4.h>
int z4date(char *date);
```

### Input

Address of field to return the date of the ZIP+4 database. This field must be at least nine bytes in length.

### Output

The date of the ZIP+4 database. This field must be at least nine bytes in length.

### Return

- 0 - Success
- 1 - Failure
- 2 - The USPS Address Matching System not ready

### Example

```
#include <stdio.h>
#include <stdlib.h>
#include <zip4.h>

char date[9];

int main(int argc, char** argv)
{
    Z4OPEN_PARM openparm;

    memset (&openparm, 0, sizeof(openparm));

    /* ... Populate openparm ... */

    /* open the USPS Address Matching System */
    if (z4opencfg(&openparm) != 0)
    {
        printf("The USPS Address Matching System failed to open");

        /* Always call z4close() even on open failure */
        z4close();

        exit(5);
    }

    /* get release date */
    z4date(date);

    printf("Release date: %s\n", date);

    /* close The USPS Address Matching System */
```

```
    z4close();  
    exit(0);  
}
```

## Get AMS Data Expiration

The `z4GetDataExpireDays()` (Get AMS Data Expiration) function instructs the Address Matching System to return the number of days until the AMS database expires. Because the function can be used periodically to check the number of days remaining until database expiration, it is strongly recommended that you integrate this function into your software.

*Note:* This function replaces the `z4expire()` function.

### Syntax

```
#include <zip4.h>
int z4GetDataExpireDays(void);
```

### Input

None

### Output

None

### Return

-1 – The AMS database has expired. Otherwise, the number of days until the AMS database expires.

### Example

```
#include <stdio.h>
#include <stdlib.h>
#include <zip4.h>

int main(int argc, char** argv)
{
    Z4OPEN_PARM openparm;

    int days = 0;

    memset (&openparm, 0, sizeof(openparm));

    /* ... Populate openparm ... */

    /* open the USPS Address Matching System */
    if (z4opencfg(&openparm) != 0)
    {
        printf("The USPS Address Matching System failed to open");

        /* Always call z4close() even on open failure */
        z4close();

        exit(5);
    }

    /* get number of days until database expiration */
    days = z4GetDataExpireDays();

    if (days == -1)
        printf("AMS database has already expired.\n");
}
```

```
else
    printf("%d days until AMS database expiration.\n", days);

/* close The USPS Address Matching System */
z4close();
}
```

## Get AMS Library Expiration

The `z4GetCodeExpireDays()` (Get AMS Library Expiration) function instructs the Address Matching System to return the number of days until the AMS library expires. Because the function can be used periodically to check the number of days remaining until library expiration, it is strongly recommended that you integrate this function into your software.

### Syntax

```
#include <zip4.h>

int z4GetCodeExpireDays(void);
```

### Input

None

### Output

None

### Return

-1 – The AMS library has expired. Otherwise, the number of days until the AMS library expires.

### Example

```
#include <stdio.h>
#include <stdlib.h>
#include <zip4.h>

int main(int argc, char** argv)
{
    Z4OPEN_PARM openparm;

    int days = 0;

    memset (&openparm, 0, sizeof(openparm));

    /* ... Populate openparm ... */

    /* open the USPS Address Matching System */
    if (z4opencfg(&openparm) != 0)
    {
        printf("The USPS Address Matching System failed to open");

        /* Always call z4close() even on open failure */
        z4close();

        exit(5);
    }

    /* get number of days until database expiration */
    days = z4GetCodeExpireDays();

    if (days == -1)
        printf("AMS library has already expired.\n");
    else
        printf("%d days until AMS library expiration.\n", days);
}
```

```
/* close The USPS Address Matching System */  
z4close();  
}
```

## Get CD-ROM Expiration Information – [Deprecated]

The `z4expire()` (Get CD-ROM Expiration Information) function instructs the Address Matching System to return the number of days until the CD-ROM expires. Because the function can be used periodically to check the number of days remaining until CD-ROM expiration, it is strongly recommended that you integrate this function into your software.

**Note:** *The `z4expire()` function is maintained for continued compatibility with existing software. However, it is deprecated and all software should begin to replace all occurrences of `z4expire()` with `z4GetDataExpireDays()`.*

### Syntax

```
#include <zip4.h>
int z4expire(void);
```

### Input

None

### Output

None

### Return

-1 - CD-ROM has expired. Otherwise, the function returns the number of days until CD-ROM expiration.

### Example

```
#include <stdio.h>
#include <stdlib.h>
#include <zip4.h>

int main(int argc, char** argv)
{
    Z4OPEN_PARM openparm;

    int days = 0;

    memset (&openparm, 0, sizeof(openparm));

    /* ... Populate openparm ... */

    /* open the USPS Address Matching System */
    if (z4opencfg(&openparm) != 0)
    {
        printf("The USPS Address Matching System failed to open");

        /* Always call z4close() even on open failure */
        z4close();

        exit(5);
    }

    /* get number of days until CD-ROM expiration */
    days = z4expire();
    if (days == -1)
        printf("CD-ROM has already expired.\n");
}
```

```
else
    printf("%d days until CD-ROM expiration.\n", days);

/* close The USPS Address Matching System */
z4close();

exit(0);
}
```

## Get API Code Version

The `z4ver()` (Get API Code Version) function commands the program to retrieve the version string of the API code. This string is in compliance with the CASS requirements for address matching software version information and may be used when generating a PS Form 3553 for mailing discounts.

**Note:** *Most functions require you to call `z4opencfg()` first to initialize the AMS system. This function does not require the AMS system to be open.*

### Syntax

```
#include <zip4.h>
int z4ver(char *str);
```

### Input

none

### Output

`str` pointer to data buffer to receive the string

### Return

0 - Success

### Example

```
#include <stdio.h>
#include <zip4.h>

void main(void)
{
    char version[32];

    /* get the Address Matching System version */
    z4ver(version) ;

    printf("The Address Matching System version is %s\n", version) ;
    exit (0);
}
```

## Multiple Response Stack

### Scroll the Stack of Address Records

The `z4scroll()` (Scroll the Stack of Address Records) function commands the Address Matching System to access additional stacks of ten address records each. The function is related to the `z4adring()` and `z4xrfinq()` functions, which return up to ten records when the `Z4_MULTIPLE` or `Z4_DEFAULT` return codes are set. When the `parm.respn` field contains a number greater than ten, your program can use this function to obtain additional stacks of ten address records (up to the number of records specified in the `parm.respn` return field). This function may only be called immediately after a call to the `z4adring()` or `z4xrfinq()` functions.

### Syntax

```
#include <zip4.h>

int z4scroll(parm);
```

### Input

The `parm` argument must point to a `ZIP4_PARM` structure. This structure should not be modified after the call to `z4adring()`.

### Output

The `parm.stack` field will be updated to contain the next ten records (fewer records may be returned if less than ten records remain).

### Return

- 0 - Success
- 1 - The USPS Address Matching System not installed
- 2 - The USPS Address Matching System not open
- 3 - Stack access not allowed

### Example

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <zip4.h>

ZIP4_PARM parm;

int main(int argc, char** argv)
{
    Z4OPEN_PARM openparm;

    int i = 0;

    memset (&openparm, 0, sizeof(openparm));

    /* ... Populate openparm ... */

    /* open the USPS Address Matching System */
    if (z4opencfg(&openparm) != 0)
    {
        printf("The USPS Address Matching System failed to open");
    }
}
```

```

        /* Always call z4close() even on open failure */
        z4close();

        exit(5);
    }

    /* Create parameter list and call the USPS Address Matching System */
    memset(&parm, 0, sizeof(parm));

    strcpy(parm.iadl1, "1336 CHATMAN"      );
    strcpy(parm.ictyi, "CORDOVA TN"      );

    z4adrinq(&parm);

    /* process all addresses returned by The USPS Address Matching System */
    for(i=0; i<parm.respn; i++)
    {
        /* check if stack needs to be refreshed */
        if (i != 0 && (i% 10) == 0)
        {
            if(z4scroll(&parm))
                break;
        }

        /* examine each address returned by The USPS Address Matching System */
        ...
    }

    /* close The USPS Address Matching System */
    z4close();

    exit(0);
}

```

## Get Last Error

The `z4geterror()` (Get Last Error) function retrieves the last error that was encountered after a failed `z4opencfg()` function call.

### Syntax

```
#include <zip4.h>
int z4geterror(Z4_ERROR *pError);
```

### Input

Pointer to an empty `Z4_ERROR` structure.

### Output

`pError` will be populated with the last error that was encountered.

### Return

The value of `iErrorCode`

#defines for the `iErrorCode` values and their meanings:

<code>ERROR_FILE_OPEN</code>	1	Error opening a file
<code>ERROR_FILE_READ</code>	2	Error reading a file
<code>ERROR_FILE_WRITE</code>	3	Error writing to a file
<code>ERROR_FILE_FIND</code>	4	Error finding a file
<code>ERROR_FILE_EXPIRE</code>	5	AMS library has expired
<code>ERROR_FILE_SYNC</code>	6	AMS Database files are out of sync
<code>ERROR_SECURITY</code>	7	AMS Security error

### Example

```
#include <stdio.h>
#include <string.h>
#include <zip4.h>

int main(int argc, char** argv)
{
    Z4_ERROR    errorparm;
    Z4_ENV      envparm;
    Z4OPEN_PARM openparm;

    memset(&errorparm, 0, sizeof(Z4_ERROR) );
    memset(&envparm, 0, sizeof(Z4_ENV) );
    memset(&openparm, 0, sizeof(Z4OPEN) PARM));
```

```

/* ... Populate openparm ... */

/* open the USPS Address Matching System */
if(z4opencfg(&openparm) != 0)
{
    printf("The USPS Address Matching System failed to open");

    z4getenv(&envparm);
    z4geterror(&errorparm);

    /* Detailed Error Information */
    printf("\n\nDETAILED ERROR INFORMATION\n");
    printf("-----\n");
    printf("Error Message:  %s\n", errorparm.strErrorMessage);
    printf("File Name:      %s\n", errorparm.strFileName);
    printf("Diagnostics:    %s\n", errorparm.strDiagnostics);

    /* Detailed Environment Information */
    printf("\n\nDETAILED ENVIRONMENT INFORMATION\n");
    printf("-----\n");
    printf("Configuration File:  %s\n", envparm.strConfigFile);
    printf("Address1:           %s\n", envparm.address1);
    printf("AddrIndex:          %s\n", envparm.addrindex);
    printf("CityState:          %s\n", envparm.citystate);
    printf("CrossRef:           %s\n", envparm.crossref);
    printf("System:             %s\n", envparm.system);
    printf("eLOT:               %s\n", envparm.elot);
    printf("eLOTIndex:          %s\n", envparm.elotindex);
    printf("EWS Path:           %s\n", envparm.ewspath);
    printf("eLOT Flag:          %s\n", envparm.elotflag);
}
else
{
    printf("The USPS Address Matching System opened successfully\n");
}
z4close();
return 0;
}

```

## Get Environment

The `z4getenv()` (Get Environment) function retrieves the environment for the Address Matching System.

### Syntax

```
#include <zip4.h>
int z4getenv(Z4_ENV *pEnv);
```

### Input

Pointer to an empty `Z4_ENV` structure.

### Output

`pEnv` will be populated with the environment for the Address Matching System.

### Return

0 - Success

### Example

See example for function `z4geterror()` (page 52).

## Retrieving the LACSLink Security Key

The `z4LLkGetKey()` function returns the stop processing security key used to disable LACSLink. A stop processing security key is an alphanumeric character string that is randomly generated when a LACSLink security violation occurs.

You may call the `z4LLkGetKey()` function after a LACSLink security violation occurs. In order to identify a LACSLink security violation, check for a return value of 7 (seven) after making an open call. At that point you may call `z4LLkGetKey()` to retrieve the randomly generated stop processing security key.

The stop processing security key returned from `z4LLkGetKey()` will be used to generate the corresponding enable security key you need for `z4LLkSetKey()`. You can obtain an enable security key from a customer care representative in exchange for the stop processing security key given to you by `z4LLkGetKey()`.

**Note:** *During a `z4LLkGetKey()` call OS resources are allocated so a call to `z4close()` must be made in order to free the resources.*

### Syntax

```
#include <zip4.h>
const char* Z4FUNC z4LLkGetKey(void);
```

### Input

none

### Output

none

### Return

const char\* - pointer to a null terminated alphanumeric character string

### Example

```
#include <stdio.h>
#include <zip4.h>

void main( void)
{
    Z4OPEN_PARM OpenParm;
    char        szKey[32] = {0};
    int         iReturn   = 0;

    memset(&openparm, 0, sizeof(Z4OPEN_PARM));

    /* Setting up paths */
    OpenParm.config.address1 = "c:\\amsdata\\";
    OpenParm.config.addrindex = "c:\\amsdata\\";
    OpenParm.config.cdrom    = "d:\\\";
```

```

OpenParam.config.citystate = "c:\\amsdata\\";
OpenParam.config.crossref  = "c:\\amsdata\\";
OpenParam.config.system    = "c:\\amsdata\\";
OpenParam.config.llkpath   = "c:\\llkdata\\";
OpenParam.config.dpvpath   = "c:\\dpvdata\\";

/* open the USPS Address Matching System */
iReturn = z4opencfg(&OpenParam);

/* success */
if( iReturn == 0 )
{
    printf("\nThe USPS Address Matching System Opened
Successfully.");
}

/* LACSLink security violation */
else if( iReturn == 7 )
{
    const char* szCode = z4LLkGetKey();

    /* display error message an security code */
    printf("\nLACSLink has been disabled.");
    printf("\n\nSecurity code: %s", szCode);
    printf("\nTo enable LACSLink contact customer support with the
security");
    printf("\ncode above to receive the security key you need to
enable LACSLink\n");

    /* prompt for security key */
    printf("\nEnter security key w/o formatting characters: ");
    gets(szKey);

    /* verify security key */
    if( z4LLkSetKey(szKey) )
    {
        /* inform user of success */
        printf("\nThe key %s is valid and LACSLink is enabled.",
szKey);
    }
    else
    {
        /* inform user of failure */
        printf("\nThe key %s is invalid and LACSLink is disabled.",
szKey);
    }
}
/* other errors */
else
{
    printf("\nError Opening the USPS Address Matching System.");
}

/* close the USPS Address Matching System */
z4close();
}

```



## Checking for LACSLink Functionality

The `z4LLkIsDisabled()` identifies when LACSLink functionality is enable/disabled. When the return value from `z4LLkIsDisabled()` is TRUE (non-zero) LACSLink is disabled otherwise LACSLink is enabled.

Before LACSLink can be enabled a system open call must be made with the `Z4OPEN_PARM.config.llkpath` containing the path to the LACSLink data files. After the open call you may check the state of LACSLink via `z4LLkIsDisabled()`.

**Note:** *During a `z4LLkIsDisabled()` call OS resources may be allocate so a call to `z4close()` must be made in order to free the resources.*

### Syntax

```
#include <zip4.h>
int Z4FUNC z4LLkIsDisabled(void);
```

### Input

none

### Output

none

### Return

TRUE - LACSLink is disabled

FALSE - LACSLink is enabled

### Example

```
#include <stdio.h>
#include <zip4.h>

void main( void)
{
    Z4OPEN_PARM OpenParm;
    char          szKey[32] = {0};
    int           iReturn   = 0;

    memset(&openparm, 0, sizeof(Z4OPEN_PARM));

    /* Setting up paths */
    OpenParm.config.address1 = "c:\\amsdata\\";
    OpenParm.config.addrindex = "c:\\amsdata\\";
    OpenParm.config.cdrom = "d:\\";
    OpenParm.config.citystate = "c:\\amsdata\\";
    OpenParm.config.crossref = "c:\\amsdata\\";
    OpenParm.config.system = "c:\\amsdata\\";
    OpenParm.config.llkpath = "c:\\llkdata\\";
    OpenParm.config.dpvpath = "c:\\dpvdata\\";

    /* open the USPS Address Matching System */
    iReturn = z4opencfg(&OpenParm);
```

```

/* success */
if( iReturn == 0 )
{
    printf("\nThe USPS Address Matching System Opened
Successfully.");
}

/* LACSLink security violation */
else if( iReturn == 7 )
{
    const char* szCode = z4LLkGetCode();

    /* display error message an security code */
    printf("\nLACSLink has been disabled.");
    printf("\n\nSecurity code: %s", szCode);
    printf("\nTo enable LACSLink contact customer support with the
security");
    printf("\ncode above to receive the security key you need to
enable LACSLink\n");

    /* prompt for security key */
    printf("\nEnter security key w/o formatting characters: ");
    gets(szKey);

    /* verify security key */
    if( z4LLkSetKey(szKey) )
    {
        /* inform user of success */
        printf("\nThe key %s is valid and LACSLink is enabled.",
szKey);
    }
    else
    {
        /* inform user of failure */
        printf("\nThe key %s is invalid and LACSLink is disabled.",
szKey);
    }
}
/* other errors */
else
{
    printf("\nError Opening the USPS Address Matching System.");
}

/* close the USPS Address Matching System */
z4close();
}

```

## Disabling the LACSLink Security Key

The `z4LLkSetKey()` function verifies the stop processing security key used for enabling LACSLink after a LACSLink security violation. A security key is an alphanumeric character string given to you by a customer care representative in exchange for the security code given to you by `z4LLkGetCode()`.

Make a call to `z4LLkSetKey()` after a LACSLink security violation. In order to identify a LACSLink security violation, check for a return value of 7 (seven) after making an open call. At that point you may call `z4LLkSetKey()` with the security key provided to you by a customer care representative.

A status of TRUE (non-zero) is returned to identify success (LACSLink is enabled) otherwise failure occurred (LACSLink is disabled).

**Note:** *`z4LLkSetKey()` must be called after a `z4opencfg()` function call. Even if the `z4opencfg()` function call fails to open AMS, it has put AMS in a state to be able to accept the key information.*

*Since this process causes AMS to allocate OS resources, the `z4close()` function call must be called in order to allow AMS to free those resources.*

### Syntax

```
#include <zip4.h>

int Z4FUNC z4LLkSetKey(const char* szKey);
```

### Input

const char\*      pointer to a null terminated alphanumeric character string

### Output

none

### Return

TRUE - The key update is successful and LACSLink is enabled

FALSE - The key update failed and LACSLink is disabled

### Example

```
#include <stdio.h>
#include <zip4.h>

void main( void)
{
    Z4OPEN_PARM OpenParm;
    char          szKey[32] = {0};
    int           iReturn   = 0;

    memset(&OpenParm, 0, sizeof(Z4OPEN_PARM));

    /* Setting up paths */
    OpenParm.config.address1 = "c:\\amsdata\\";
    OpenParm.config.addrindex = "c:\\amsdata\\";
    OpenParm.config.ctystate = "c:\\amsdata\\";
    OpenParm.config.crossref = "c:\\amsdata\\";
    OpenParm.config.system = "c:\\bin\\";
```

```

OpenParm.config.llkpath      = "c:\\llkdata\\";
OpenParm.config.dpvpath     = "c:\\dpvdata\\";

/* open the USPS Address Matching System */
iReturn = z4opencfg(&OpenParm);

/* success */
if( iReturn == 0 )
{
    printf("\nThe USPS Address Matching System Opened
Successfully.");
}
/* LACSLink security violation */
else if( iReturn == 7 )
{
    const char* szCode = z4LLkGetCode();

    /* display error message an security code */
    printf("\nLACSLink has been disabled.");
    printf("\n\nSecurity code: %s", szCode);
    printf("\nTo enable LACSLink contact customer support with the
security");
    printf("\ncode above to receive the security key you need to
enable LACSLink\n");

    /* prompt for security key */
    printf("\nEnter security key w/o formatting characters: ");
    gets(szKey);

    /* verify security key */
    if( z4LLkSetKey(szKey) )
    {
        /* inform user of success */
        printf("\nThe key %s is valid and LACSLink is enabled.",
szKey);
    }
    else
    {
        /* inform user of failure */
        printf("\nThe key %s is invalid and LACSLink is disabled.",
szKey);
    }
}
/* all other errors */
else
{
    printf("\nError Opening the USPS Address Matching System.");
}

/* close the USPS Address Matching System */
z4close();
}

```

### **Section 3: Footnote Flags**

**A ZIP CODE CORRECTED**

The address was found to have a different 5-digit ZIP Code than given in the submitted list. The correct ZIP Code is shown in the output address.

**B CITY / STATE SPELLING CORRECTED**

The spelling of the city name and/or state abbreviation in the submitted address was found to be different than the standard spelling. The standard spelling of the city name and state abbreviation are shown in the output address.

**C INVALID CITY / STATE / ZIP**

The ZIP Code in the submitted address could not be found because neither a valid city, state, nor valid 5-digit ZIP Code was present. It is also recommended that the requestor check the submitted address for accuracy.

**D NO ZIP+4 ASSIGNED**

This is a record listed by the United States Postal Service on the national ZIP+4 file as a non-deliverable location. It is recommended that the requestor verify the accuracy of the submitted address.

**E ZIP CODE ASSIGNED FOR MULTIPLE RESPONSE**

Multiple records were returned, but each shares the same 5-digit ZIP Code.

**F ADDRESS COULD NOT BE FOUND IN THE NATIONAL DIRECTORY FILE DATABASE**

The address, exactly as submitted, could not be found in the city, state, or ZIP Code provided. It is also recommended that the requestor check the submitted address for accuracy. For example, the street address line may be abbreviated excessively and may not be fully recognizable.

**G INFORMATION IN FIRM LINE USED FOR MATCHING**

Information in the firm line was determined to be a part of the address. It was moved out of the firm line and incorporated into the address line.

**H MISSING SECONDARY NUMBER**

ZIP+4 information indicates this address is a building. The address as submitted does not contain an apartment/suite number. It is recommended that the requestor check the submitted address and add the missing apartment or suite number to ensure the correct Delivery Point Barcode (DPBC).

**I INSUFFICIENT / INCORRECT ADDRESS DATA**

More than one ZIP+4 Code was found to satisfy the address as submitted. The submitted address did not contain sufficiently complete or correct data to determine a single ZIP+4 Code. It is recommended that the requestor check the address for accuracy and completeness. For example, firm name, or institution name, doctor's name, suite number, apartment number, box number, floor number, etc. may be missing or incorrect. Also pre-directional or post-directional indicators (North = N, South = S, East = E, West = W, etc.) and/or street suffixes (Street = ST, Avenue = AVE, Road = RD, Circle = CIR, etc.) may be missing or incorrect.

- J DUAL ADDRESS**  
The input contained two addresses. For example: 123 MAIN ST PO BOX 99.
- K MULTIPLE RESPONSE DUE TO CARDINAL RULE**  
CASS rule does not allow a match when the cardinal point of a directional changes more than 90%.
- L ADDRESS COMPONENT CHANGED**  
An address component (i.e., directional or suffix only) was added, changed, or deleted in order to achieve a match.
- M STREET NAME CHANGED**  
The spelling of the street name was changed in order to achieve a match.
- N ADDRESS STANDARDIZED**  
The delivery address was standardized. For example, if STREET was in the delivery address, the system will return ST as its standard spelling.
- P BETTER ADDRESS EXISTS**  
The delivery address is matchable, but is known by another (preferred) name. For example, in New York, NY, AVENUE OF THE AMERICAS is also known as 6TH AVE. An inquiry using a delivery address of 55 AVE OF THE AMERICAS would be flagged with a Footnote Flag P.
- Q UNIQUE ZIP CODE MATCH**  
Match to an address with a unique ZIP Code.
- R NO MATCH DUE TO EWS**  
The delivery address is matchable, but the EWS file indicates that an exact match will be available soon.
- S INCORRECT SECONDARY ADDRESS**  
The secondary information (i.e., floor, suite, apartment, or box number) does not match that on the national ZIP+4 file. This secondary information, although present on the input address, was not valid in the range found on the national ZIP+4 file.
- T MULTIPLE RESPONSE DUE TO MAGNET STREET SYNDROME**  
The search resulted in a single response; however, the record matched was flagged as having magnet street syndrome. “Whenever an input address has a single suffix word or a single directional word as the street name, or whenever the ZIP+4 File records being matched to have a single suffix word or a single directional word as the street name field, then an exact match between the street, suffix and/or post-directional and the same components on the ZIP+4 File must occur before a match can be made. Adding, changing or deleting a component from the input address to obtain a match to a ZIP+4 record will be considered incorrect.” Instead of returning a “no match” in this situation a multiple response is returned to allow access the candidate record.
- U UNOFFICIAL POST OFFICE NAME**  
The city or post office name in the submitted address is not recognized by the United States Postal Service as an official last line name (preferred city name), and is not acceptable as an alternate name. This does denote an error and the preferred city name will be provided as output.
- V UNVERIFIABLE CITY / STATE**  
The city and state in the submitted address could not be verified as corresponding to the given 5-digit ZIP Code. This comment does not necessarily denote an error; however, it is recommended that the requestor check the city and state in the submitted address for accuracy.

- W**     **INVALID DELIVERY ADDRESS**  
The input address record contains a delivery address other than a PO BOX, General Delivery, or Postmaster with a 5-digit ZIP Code that is identified as a “small town default.” The United States Postal Service does not provide street delivery for this ZIP Code. The United States Postal Service requires use of a PO BOX, General Delivery, or Postmaster for delivery within this ZIP Code.
- X**     **UNIQUE ZIP CODE GENERATED**  
Default match inside a unique ZIP Code.
- Y**     **MILITARY MATCH**  
Match made to a record with a military ZIP Code.
- Z**     **MATCH MADE USING THE ZIPMOVE PRODUCT DATA**  
The ZIPMOVE product shows which ZIP + 4 records have moved from one ZIP Code to another. If an input address matches to a ZIP + 4 record which the ZIPMOVE product indicates as having moved, the search is performed again in the new ZIP Code.

## **Section 4: Record Types**

**F FIRM**

This is a match to a Firm Record, which is the finest level of match available for an address.

**G GENERAL DELIVERY**

This is a match to a General Delivery record.

**H BUILDING / APARTMENT**

This is a match to a Building or Apartment record.

**P POST OFFICE BOX**

This is a match to a Post Office Box.

**R RURAL ROUTE or HIGHWAY CONTRACT**

This is a match to either a Rural Route or a Highway Contract record, both of which may have associated Box Number ranges.

**S STREET RECORD**

This is a match to a Street record containing a valid primary number range.

## Section 4: Return Codes

- 10 INVALID DUAL ADDRESS**  
Information presented could not be processed in current format. Corrective action is needed. Be sure that the address line components are correct. For example, the input address line may contain more than one delivery address.
- 11 INVALID CITY/ST/ZIP**  
The ZIP Code in the submitted address could not be found because neither a valid city, state, nor valid 5-digit ZIP Code was present. Corrective action is needed. It is also recommended that the requestor check the submitted address for accuracy.
- 12 INVALID STATE**  
The state in the submitted address is invalid. Corrective action is needed. It is also recommended that the requestor check the submitted address for accuracy.
- 13 INVALID CITY**  
The city in the submitted address is invalid. Corrective action is needed. It is also recommended that the requestor check the submitted address for accuracy.
- 21 NOT FOUND**  
The address, exactly as submitted, could not be found in the national ZIP+4 file. It is recommended that the requestor check the submitted address for accuracy. For example, the street address line may be abbreviated excessively and may not be fully recognizable.
- 22 MULTIPLE RESPONSE**  
More than one ZIP+4 Code was found to satisfy the address submitted. The submitted address did not contain sufficiently complete or correct data to determine a single ZIP+4 Code. It is recommended that the requestor check the address for accuracy and completeness. Address elements may be missing
- 31 EXACT MATCH**  
Single response based on input information. No corrective action is needed since an exact match was found in the national ZIP+4 file.
- 32 DEFAULT MATCH**  
A match was made to a default record in the national ZIP+4 file. A more specific match may be available if a secondary number (i.e., apartment, suite, etc.) exists.

## Appendix A: Interface Definition

```

#ifndef ZIP4_H                /* avoid redefinition */
#define ZIP4_H
/*****
/* This record describes an address record. The record format is */
/* the same as the USPS ZIP+4 File. Please see the USPS Address */
/* Information Products Technical Guide for information on this record. */
/* NOTE:All 'char' array fields contain an extra byte (+1) for the null */
/* terminator. */
*****/
typedef struct
{
    char    detail_code;        /* copyright detail code */
    char    zip_code[5+1];     /* zip code */
    char    update_key[10+1];  /* update key number */
    char    action_code;       /* action code */
    char    rec_type;          /* record type */
    char    carr_rt[4+1];      /* carrier route */
    char    pre_dir[2+1];      /* pre-direction abbrev */
    char    str_name[28+1];    /* street name */
    char    suffix[4+1];       /* suffix abbrev */
    char    post_dir[2+1];     /* post-direction abbrev */
    char    prim_low[10+1];    /* primary low range */
    char    prim_high[10+1];   /* primary high range */
    char    prim_code;         /* primary even odd code */
    char    sec_name[40+1];    /* bldg or firm name */
    char    unit[4+1];         /* secondary abbreviation */
    char    sec_low[8+1];      /* secondary low range */
    char    sec_high[8+1];     /* secondary high range */
    char    sec_code;          /* secondary even odd code */
    char    addon_low[4+1];    /* add on low */
    char    addon_high[4+1];   /* add on high */
    char    base_alt_code;     /* base alternate code */
    char    lacs_status;       /* LACS converted status */
    char    finance[6+1];      /* finance code */
    char    state_abbrev[2+1]; /* state abbreviation (not filled) */
    char    county_no[3+1];    /* county number */
    char    congress_dist[2+1]; /* congressional district */
    char    municipality[6+1]; /* municip. city/state key (not filled)*/
    char    urbanization[6+1]; /* urb. city/state key */
    char    last_line[6+1];    /* last line city/state key */
} ADDR_REC;

/* NOTE: The GovtBldgInd (Government Building Indicator) field is not*/
/* available in the ADDR_REC structure. */

```

```

/*****
/*   This record describes a city/state record. The record format is the   */
/*   same as the USPS City State File. Please see the USPS Address       */
/*   Information Products Technical Guide for information on this record.   */
/*   NOTE: All 'char' array fields contain an extra byte (+1) for the     */
/*   null terminator.                                                     */
/*****
typedef struct
{
    char    detail_code;           /* copyright detail code           */
    char    zip_code[5+1];        /* zip code                         */
    char    city_key[6+1];        /* city/state key                   */
    char    zip_class_code;       /* zip classification code         */
                                    /* blank = non-unique zip          */
                                    /* M=APO/FPO military zip         */
                                    /* P=PO BOX zip                    */
                                    /* U=Unique zip                     */
    char    city_name[28+1];      /* city/state name                  */
    char    city_abbrev[13+1];    /* city/state name abbrev          */
    char    facility_cd;         /* facility code                    */
                                    /* A=Airport mail facility        */
                                    /* B=Branch                        */
                                    /* C=Community post office        */
                                    /* D=Area distrib. center         */
                                    /* E=Sect. center facility        */
                                    /* F=General distrib. center     */
                                    /* G=General mail facility        */
                                    /* K=Bulk mail center             */
                                    /* M=Money order unit            */
                                    /* N=Non-postal name              */
                                    /* community name,                */
                                    /* former postal facility,        */
                                    /* or place name                  */
                                    /* P=Post office                  */
                                    /* S=Station                      */
                                    /* U=Urbanization                 */
    char    mailing_name_ind;     /* mailing name indicator          */
                                    /* Y=Mailing name                  */
                                    /* N=Non-mailing name             */
    char    last_line_num[6+1];   /* preferred last line key         */
    char    last_line_name[28+1]; /* preferred city name             */
    char    city_delv_ind;        /* city delivery indicator         */
                                    /* Y=Office has city              */
                                    /* delivery carrier rts           */
                                    /* N=Office does not have        */

```

```

char auto_zone_ind; /* city delivery carrier */
/* routes */
/* automated zone indicator */
/* A=CR Sort Rates Apply */
/* Merge Allowed */
/* B=CR Sort Rates Apply */
/* Merge Not Allowed */
/* C=CR Sort Rates Do Not Apply */
/* Merge Allowed */
/* D=CR Sort Rates Do Not Apply */
/* Merge Not Allowed */
char unique_zip_ind; /* unique zip name indicator */
/* Y=Unique zip name */
/* blank=not applicable */
char finance[6+1]; /* finance code */
char state_abbrev[2+1]; /* state abbreviation */
char county_no[3+1]; /* county number */
char county_name[25+1]; /* county name */
} CITY_REC;
/*****
/* Parameter list for z4adrinq() and z4xrfinq() calls. Reserved */
/* fields are for future use, do not access these fields. Size of his */
/* record cannot be changed. */
/* NOTE: Only fields containing +1 in the length are null terminated. */
*****/
typedef struct
{
char rsvd0[4]; /******input data******/
/* reserve fore future use */
char iadl1[50+1]; /* input delivery address */
char iadl2[50+1]; /* input firm name */
char ictyi[50+1]; /* input city */
char istai[2+1]; /* input state */
char izipc[10+1]; /* input ZIP+4 code */
char iprurb[28+1]; /* input urbanization name */
char iadl3[50+1]; /* input second address line */
char iddpv11[12+1]; /* input value for direct DPV */
char rsvd1[85]; /* reserved for future use */
/******returned data *****/
char dadl3[50+1]; /* standardized 2nd delivery address*/
char dadl1[50+1]; /* standardized delivery address */
char dadl2[50+1]; /* standardized firm name */
char dlast[50+1]; /* standardized city/state/zip */
char dprurb[28+1]; /* output PR urbanization name */
}

```

```

char  dctys[28+1];      /* main post office city      */
char  dstas[2+1];      /* main post office state     */
char  dctya[28+1];     /* standardized city          */
char  abcty[13+1];     /* standardized city abbreviation */
char  dstaa[2+1];      /* standardized state         */
char  zipc[5+1];       /* 5-digit zip code           */
char  addon[4+1];      /* ZIP+4 addon code           */
char  dpbc[3+1];       /* delivery point bar code    */
char  cris[4+1];       /* carrier route              */
char  county[3+1];     /* FIPS county code           */
short respn;          /* number of returned responses */
char  retcc;           /* return code                 */
char  adrkey[12];      /* address key (for indexing)  */
char  auto_zone_ind;   /* A, B, C or D               */
char  elot_num[4+1];   /* eLOT Number                 */
char  elot_code;       /* eLOT Ascending/Descending Flag */
char  llk_rc[2+1];     /* LACSLink Return Code       */
char  llk_ind;         /* LACSLink Indicator          */
char  misc[128+1];     /* line for unused input data  */
char  rsvd2[20];       /* Reserved for Future Use     */

                                /****** parsed input data***** */
char  ppnum[10+1];     /* Primary Number              */
char  psnum[8+1];      /* Secondary Number            */
char  prote[3+1];      /* Rural Route Number          */
char  punit[4+1];      /* Secondary Number Unit       */
char  ppre1[2+1];      /* First or Left Pre-direction */
char  ppre2[2+1];      /* Second or Right Pre-direction */
char  psuf1[4+1];      /* First or Left Suffix        */
char  psuf2[4+1];      /* Second or Right Suffix      */
char  ppst1[2+1];      /* First or Left Post-direction */
char  ppst2[2+1];      /* Second or Right Post-direction */
char  ppnam[28+1];     /* Primary Name                */
char  mpnum[10+1];     /* Matched primary number.     */
char  msnum[8+1];      /* Matched secondary number    */
char  pmb[3+1];        /* PMB Unit Designator         */
char  pmbnum[8+1];     /* PMB Number                  */
char  mlev1;           /* Reserved Use                 */

char  footnotes[32+1] /* Footnote String            */
char  rsvd3[28];       /* Reserved for Future Use     */

struct {
                                /****** footnotes***** */
char  a;               /* zip corrected               */
char  b;               /* city/state corrected        */
char  c;               /* invalid city/state/zip     */
char  d;               /* no zip assigned             */
char  e;               /* ZIP assigned for mult response */
char  f;               /* no zip available            */
char  g;               /* part of firm moved to address */

```

```

char h;          /* secondary number missing */
char i;          /* insufficient/incorrect data */
char j;          /* dual input */
char k;          /* reserved for future use" */
char l;          /* del addr component add/del/chg */
char m;          /* street name spelling changed */
char n;          /* delivery addr was standardized */
char o;          /* reserved for future use */
char p;          /* better delivery addr exists */
char q;          /* Unique ZIP Code */
char r;          /* no match caused by EWS */
char s;          /* invalid secondary number */
char t;          /* magnet street */
char u;          /* unofficial PO name */
char v;          /* unverifiable city/state */
char w;          /* small town default */
char x;          /* unique ZIP Code generated */
char y;          /* Military Match */
char z;          /* ZIP Move Match */
char rsvd3[6];  /* reserved for future use */
} foot;

ADDR_REC stack[10]; /******record stack*****/
char rsvd4[194];   /* reserved for future use */
} ZIP4_PARM;

/*****
/* Parameter list for z4getzip() */
/* NOTE: Only fields containing +1 in the length are null terminated. */
*****/

typedef struct
{
    char input_cityst[50+1];
    char output_cityst[50+1];
    char low_zipcode[5+1];
    char high_zipcode[5+1];
    char finance_num[6+1];
} GET_ZIPCODE_STRUCT;

/*****
/* Error Codes for the iErrorCode variable inside the Z4_ERROR */
/* structure */
*****/

#define ERROR_FILE_OPEN 1 /* Error opening a file */
#define ERROR_FILE_READ 2 /* Error reading a file */
#define ERROR_FILE_WRITE 3 /* Error writing to a file */
#define ERROR_FILE_FIND 4 /* Error finding a file */
#define ERROR_FILE_EXPIRE 5 /* AMS library has expired */
#define ERROR_FILE_SYNC 6 /* AMS Database files out of sync */
#define ERROR_SECURITY 7 /* AMS Security Error */

#define FILE_ID_CONFIG 1 /* Configuration File */
#define FILE_ID_ZDRFLE 2 /* zadrfile.dat */
#define FILE_ID_ZDRFLENDX 3 /* zadrfile.idx */

```

```

#define FILE_ID_CTYSTATE          4
#define FILE_ID_CTYSTATENDX      5
#define FILE_ID_ZIP5FLE          6
#define FILE_ID_ZIP5FLENDX      7
#define FILE_ID_ZXREFDTL         8
#define FILE_ID_ELTRVFLE         9
#define FILE_ID_ELTRVFLENDX     10
#define FILE_ID_EWS              11
#define FILE_ID_SYSTEM           12
#define FILE_ID_LIBRARY          13
#define FILE_ID_KEYMANLIB        14
#define FILE_ID_DATABASE         15
#define FILE_ID_LLK              16
#define FILE_ID_DPV              17
#define FILE_ID_FNSN             18

/*****
/* Parameter list for z4geterror()
/* NOTE: Only fields containing +1 in the length are null terminated
*****/
typedef struct
{
    int    iErrorCode;           /* Error Code
    char   strErrorMessage[100+1]; /* Error Message
    int    iFileCode;           /* File Code
    char   strFileName[26+1];    /* File Name
    char   strDiagnostics[300+1]; /* Diagnostic Message
} Z4_ERROR;

/*****
/* Paramter list for z4getenv()
/* NOTE: Only fields containing +1 in length are null terminated
*****/
typedef struct
{
    char   strConfigFile[300+1];
    char   address1[300+1]; /*Contains the full path of the ZADRFLE.DAT file
    char   addrindex[300+1]; /*Contains the full path of the ZADRFLE.NDX file
    char   cdrom[300+1]; /*Contains the drive letter of the CD-ROM drive that
    /*Contains the ZIP+4/carrier route data;may be blank
    char   citystate[300+1]; /*Contains the full path of the following files:
    /*CTYSTATE.DAT - CITYSTATE.NDX
    /*ZIP5FLE.DAT - ZIP5FLE.NDX
    char   crossref[300+1]; /*Contains full path of the ZXREFDTL.DAT file
    char   system[300+1]; /*Contains the full path of the Z4CXLOG.DAT file
    char   elot[300+1]; /*Contains the full path of the eltrvfle.dat file
    char   elotindex[300=1]; /*Contains the full path of the eltrvfle.ndx file
    char   llkpath[300+1] /*Contains the full path of the LACSLink files
    char   ewspath[300+1]; /*Contains the full path of the ews.txt file
    char   fnsnpath[300+1]; /*Contains the full path of the fnsn.* files
    char   rsvd1[300]; /*reserved for future use
    char   ewsflag /*EWS indicator
    char   elotflag;
    char   llkflag;
    char   dpvflag;
}Z4_ENV;

```

```

/*****
/* Parameter list for z4opencfg()
/* NOTE: Only fields containing +1 in the length are null terminated.
/*****

/*Use of this structure will replace a physical copy of the configuration
/*file on the hard drive

typedef struct
{
    char *address1; /*Contains the full path of the ZADRFLE.DAT file
    char *addrindex; /*Contains the full path of the ZADRFLE.NDX file
    char *cdrom; /*Contains the drive letter of the CD-ROM drive that
/*contains the ZIP+4/carrier route data;may be blank*/
    char *citystate; /*Contains the full path of the following files:
/*CTYSTATE.DAT - CTYSTATE.NDX
/*ZIP5FILE.DAT - ZIP5FLE.NDX
    char *crossref; /*Contains the full path of the ZXREFDTL.DAT file
    char *system; /*Contains the full path of the Z4CXLOG.DAT file
    char *elot; /*Contains the full path of the ELTRVFLE.DAT file
    char *elotindex; /*Contains the full path of the ELTRVFLE.ND file
    char *llkpath; /*Contains the full path of the LACSLink files
    char *ewspath; /*Contains the full path of the EWS.TXT file
    char *dpvpath /*Contains the full path of the dpv files
    char *fnsnpath; /*Contains the full path of the fnsn.* files
    char rsvd[124] /*Reserved for future use
}CONFIG_PARM;

typedef struct
{
    char rsvd1[50]; /*reserved for future use
    short status; /*1 - Used value point to by fname
/*2 - Used values in CONFIG_PARM
/*9 - No values found. Search for z4config.dat
    char *fname; /*pointer to a NULL terminated string that
/*contains the full path and filename for a custom*/
/*config file. If fname contains a leading space
/*or NULL then it is ignored and the CONFIG_PARM
/*is evaluated for path names
    CONFIG_PARM config; /*Contains the path name for the config file
    char ewsflag; /*Y Enabled EWS else Disable EWS
    char elotflag; /*Y Enables LOT else Disable eLOT
    char llkflag; /*Y Enables LACSLink else disable LACSLink
    char dpvflag; /*Y Enables DPV else disable DPV
    char systemflag; /*Indicates open option
    char rsvd2[511]; /*reserved for future use
}Z4OPEN_PARM

/*****
/*Z4OPEN_PARM.status values for z4opencfg()
/*****

#define Z4_FNAME 1 /*Used the value in fname as the path and filename
#define Z4_CONFIG 2 /*Used the paths in the CONFIG_PARM structure
#define Z4_SEARCH 9 /*Used neither, searched for z4config.dat

```

```

/*****
/*      Return Codes for z4adrinq() and z4xrfinq() calls      */
/*****
#define      Z4_INVADDR  10      /* invalid address      */
#define      Z4_INVZIP   11      /* invalid ZIP Code     */
#define      Z4_INVSTATE 12      /* invalid state code   */
#define      Z4_INVCITY  13      /* invalid city         */
#define      Z4_NOTFND   14      /* address not found    */
#define      Z4_MULTIPLE 22      /* multiple response - no default */
#define      Z4_SINGLE   31      /* single response - exact match */
#define      Z4_DEFAULT  32      /* default response    */

/*****
/*      Function prototypes for the ZIP+4 retrieval engine.      */
/*****
#if defined(OS2_32)
#define Z4FUNC
#elif defined(WIN32)
#define Z4FUNC _cdecl
#elif defined(_WINDOWS) || defined(_WINDLL)
#define Z4FUNC __far __pascal __export
#elif defined(OS2)
#define Z4FUNC _far _pascal _loadds _export
#elif defined(_MAC)
#define Z4FUNC
#elif defined(ANSI_STRICT) || defined(UNIX) || defined(I370)
#define Z4FUNC
#else
#define Z4FUNC _cdecl
#endif

int      Z4FUNC z4ready(void);          /* check presence of retrieval engine */
int      Z4FUNC z4remove(void);         /* terminate the retrieval engine     */
int      Z4FUNC z4open(void);           /* open the retrieval engine for use  */
int      Z4FUNC z4opencfg(Z4OPEN_PARM *); /*open with custom parameters        */
int      Z4FUNC z4close(void);          /* close the retrieval engine         */
int      Z4FUNC z4abort(void);          /* abort the current inquiry          */
int      Z4FUNC z4adrinq(ZIP4_PARM *);  /* address inquiry                    */
int      Z4FUNC z4scroll(ZIP4_PARM *);  /* address inquiry                    */
int      Z4FUNC z4adrkey(ZIP4_PARM *);  /* address key (for indexing)         */
int      Z4FUNC z4xrfinq(ZIP4_PARM *);  /* nine digit cross reference inquiry */
int      Z4FUNC z4adrstd(ZIP4_PARM *, int); /* address standardization            */

```

```

int    Z4FUNC  z4ctyget(CITY_REC *, void *); /* get first city for a state      */
int    Z4FUNC  z4ctynxt(CITY_REC *);        /* get next city for a state      */
int    Z4FUNC  z4adrget(ADDR_REC *, void *); /* get first address for a fin. no */
int    Z4FUNC  z4adrnxt(ADDR_REC *);        /* get next address for a fin. no */
int    Z4FUNC  z4date(char *);              /* get date of ZIP+4 database     */
int    Z4FUNC  z4GetDataExpireDays(void);   /* number of days until data expire */
int    Z4FUNC  z4GetCodeExpireDays(void);   /* number of days until code expire */
int    Z4FUNC  z4expire(void);              /* Deprecated. Use GetDataExpireDays() */
int    Z4FUNC  z4getzip(GET_ZIPCODE_STRUCT*); /* get zip code range for cityst */
int    Z4FUNC  z4ver(char *);               /* get the version of the API code */
int    Z4FUNC  z4geterror(Z4_ERROR *);      /* get the last error msg and code */
int    Z4FUNC  z4getenv(Z4_ENV *);          /* get the environment for AMS    */
const char* Z4FUNC  z4LlkGetKey(void);
int    Z4FUNC  z4LlkIsDisabled(void);
int    Z4FUNC  z4LlkSetKey(const char* szKey);

#endif /* ZIP4_H */

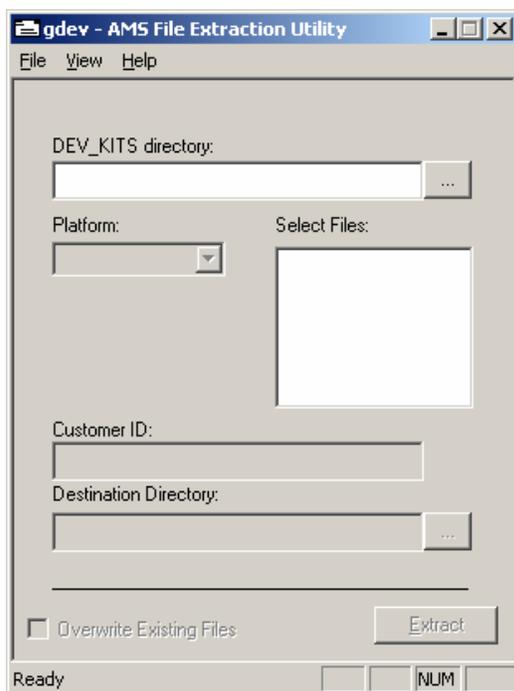
```

## Appendix B: GDEV Application

GDEV is a GUI Windows application that provides the capability to unencrypt any of the developer kits from the AMS CD-ROM.

This application is located on the AMS CD in the dev\_kits sub-directory. (gdev.exe)

If GDEV is launched directly from the CD-ROM it will automatically load information from that CD-ROM. Otherwise, GDEV will not display any information and you will have to manually select a DEV\_KITS directory.



1. Select "File->Select DEV\_KIT directory" to tell GDEV where the developer kits are located.
2. Select the "Platform" that you want to unencrypt.
3. Select the file(s) that you want to unencrypt.
4. Enter your customer ID.
5. Enter or Select the directory where the unencrypted files should be placed.
6. Click the [Extract] button.

*Note: If the "Destination Directory" already contains the selected files then you must also select the "Overwrite Existing Files" checkbox or the unencryption process will fail.*