All My Papers

AX9LIB™

Software Development Tool Kit
User Reference Guide

Version 6.2.9
August 2014

040-100044

CREDITS

AX9LIB design and programming by Larry Krummel and Brett Nelson
This manual written and produced by Joan Johnson, Larry Krummel and Brett Nelson.

All_My_Papers
13750 Serraoaks
Saratoga, CA 95070
(408) 366-6400
www.allmypapers.com

Contents

Introduction 7

General Information 8

Summary of Updates to Version 6.2.9 Manual August – 2014 9
Summary of Updates to Version 6.2.3 Manual Oct. – 2013 10
Summary of Updates to Version 2.4.5 Manual May - 2011 13
Summary of Updates to Version 2.4.4 Manual August - 2010 13
Summary of Updates to Version 2.4.3 Manual – March 2010 14
Summary of Updates to Version 2.4.2 Manual – January 2010 14
Summary of Updates to Version 2.4.1 Manual – November 2009 15
Summary of Updates to Version 2.4 Manual – September 2009 16
Summary of Updates to Version 2.3 Manual – May 2009 17
Summary of Updates to Version 2.2 Manual – October 2007 18
Summary of Updates to Version 2.1 Manual – June 2007 19
Summary of Major Updates to Version 2.0 Manual – Dec. 2006 20
Working Directory 22
Input Files 22
Output Files 23
ASCII File – Selection Lists 24
List Versus Item 26
Append/Prepend - Examples of Insertion 26
X9.37 Record Types 27
Item Detail Record Content 28
Deposit Slips 28
Valid ABA Numbers 29
Control Records 29
Date Format 30

AX9DEMO - Dashboard 30

Endorsement Screen Parameters 31
Destination (Credit)Screen Parameters 32
Sort Screen parameters 33
Summary of AX9 Screen Functions 34
AX9Lib – AX9DEMO Screen Definitions (Input Fields) 38
AX9LIB Functionality Test 48
Read/Write Functions 48
Convert From Nsf (Read Nsf/Sit Write 937) 48
Convert From 9.37 (Read X9.37 and Write X9.37) 54

Convert From CSV (Read CSV Write X9.37) 64
Convert To SOP (Read X9.37 Write SOP) 64
Convert to CSV/SIT 65
Convert to Nsf/Sit (Read 937 Write Nsf/Sit) 66
Convert To CSV/Blob (Read X9.37 Write CSV) 67
Convert To NSF/Blob (Read 937 Write NSF) 67
Convert to 180 (Read X9.37 Write 180) 68
Convert to MAL (Read X9.37 Write MAL) 69
Convert From MAL (Read MAL Write X9.37) 69
Convert To UCD (Read X937 Write UCD (187) 70
Editing Functions 71
List Copy 71
Copy a list of items into a new file – file based) 71
List Delete 72
Item Copy (Copy an Item into a new X9.37 file) 73
Item Delete 74
Split Functions 75
Split (Split by Bundle – new file for each Bundle) 75
Split Cash Letter (New file for each Cash Letter) 76
Append/Prepend Functions 77
Append 9.37 78
Append CSV – Puts CSV items in a X9.37 file 80
Prepend 9.37 81
Prepend CSV 82
Balance Functions – Balance the Control Records 83
Test Balance 83
Balance – Balance File, Cash Letters, or Bundles 84
Balance Summary File – Output files from Balance Screens (not a screen) 85
Sample Summary File 85
Field Functions 86
Get Field 86
Get Field Optimize 89
Set Field 91
Set All Fields 94
Select Functions 98
Select List 9.37 98
Select List CSV 99
Select Item 9.37 100
Select Item CSV 101
Find Pick 9.37 102
Sort/Merge Functions (Routing) 103
Sort 104
SortItems 105
SortBoFD 105
SortBoFD Account 107
Sort Payor 108
Group Account 110
Sort/Detect Dups 112
Sort Dups (No Amount) 114
Sort Micr Fields 114

Cutter Sort....116
Other Merge Functions....117
Merge....117
Merge Cash Letters....119
Merge Bundles....119
Merge Files....120
Limit Bundles....121
Return Functions....122
Forward to Return Functions....123
Write Return ALL (new 9/09)....123
Write Return (from list of Forward Items)....124
Write Return Item – Writes a single Return Item....125
Write Return Find List – Write Forward items from a list of Return Items....126
Return to Forward Functions....127
Write Return Item – Write Forward item from a single Return Item....127
Write Return....128
Duplicate Functions....129
Test Add Dup – Update the History file....129
Test Dup....130
Image and File QA Functions....131
Test Print Ready....132
Test Exchange Ready....133
Not Correctable List....136
Miscellaneous Functions....137
MICR Parse....137
Bad Char Correction....137
Bad Char Correction....138
Get File Type....139
Get Items in File....140
Get Header in File....140
Get Version – No input parameters, just click button....141
Verify 937 Nsf/Blob – read a X9.37 file and verify MICR values....142
Endorse 937....143
Get License Availability....144

APPENDEX A - INPUT FILE FORMATS 145

NSF FILE FORMAT....147
Example 1 - NSF Input Text File....147
Example 2 - NSF Credit Record....148
Example 3 - NSF Input Text File with a “28” parameter....148
NSF2 FILE FORMAT....149
Example - NSF2 Input File....149
NSFMIT FILE FORMAT....150
Example - NSFMIT....150
NSFBLOB FILE FORMAT....151
CSV FILE FORMAT....151
Example - CSV Input Text File....152

X9.37 FILE FORMAT....154
MAL FILE FORMAT....156
Example - MAL Input Text File....156

APPENDEX B - Output FILE FORMATS 157

Appendix C - Record 61 Versions 158

Overview....158
X9.37 Does Not Have a Record 61 Defined....158
Credit Record--AMP Version 1 (First Ballot 180)....159
Credit Record--AMP Version 2....160
Credit Record--AMP Version 3....161
Credit Record--Second Ballot 180 (not used in 937)....162
Credit Record 61 - Format Conversions....163

Appendix D – Record 68 - General Format 166

Appendix E – Error Codes 167

Appendix F – Sample Reports/Logs 172

Appendix G – Test Exchange Error Definitions 175

Appendix H – .NET Interface Commands for accessing AX9LIB 177

Introduction

The AX9Lib User Reference Guide is designed to provide the *end-user* an overview of AX9Lib processing. It will provide information regarding basic design characteristics which are common to all AX9Lib functions and those that are only used for special features.

This document should be used to familiarize the end-user personnel with the information displayed on the screen and its processing features. It provides field definitions of all screen entry fields and the function(s) that use them. The User Guide also provides a description of each Function AX9Lib can perform along with an example of the parameters needed to obtain the desired result.

This document will allow learning to use AX9Lib's many powerful features as they are needed. The combination of the User Guide and the Technical Reference Manuals included with the system are designed to help the end-user get started quickly. The more you use AX9Lib, the more you'll be able to take advantage of its many unique features.

To use this document and AX9Lib, you should be familiar with general PC terminology such as directory, pathname, and file name. You should be comfortable with operations such as how to name files and create subdirectories, moving the cursor around the screen, and using the keyboard and mouse.

When you have completed reading the material in this document, you should be able to:

1) Determine the correct function for the activity you are trying to perform.
2) Determine the input fields that need to be populated to obtain the result needed.
3) Navigate the screen.
4) Utilize the various functions to provide maximum benefit to your organization.
    - Convert a CSV or NSF to a X9.37 file and conversely
    - Edit the information in a X9.37 record
    - Split files into separate bundles or cash letters
    - Add records to an existing file
    - Balance files
    - Sort/Merge items
    - Segregate Return items and create a Return file
    - Recalculate the Return file
    - Repair images too large to be accepted for exchange
    - File testing for IRD Print compliance and Exchange compliance
    - Generate a Pick file with items that fail compliance testing
    - Creation and insertion of Record 61 items on either bundle or cash letter basis
    - Conversion of X9.37 format files to X9.100-180 format (requires a X9.180 license)
    - Merge files, cash letters, and bundles
    - Generate an XML "payload" file with a summary of file contents
    - Create a Destination file and/or Report file for various functions

General Information

The General Information Section will provide the following:

Summary of the new updates to the Manual
General comments on AX9LIB processing and file structure
Differentiation between Selection Lists and Pick Lists
Examples of insertion using Append/Prepend
Definition of Control Records
Use of Deposit Slips

Note: For complete reference material on AX9LIB, see the documents included with this installation in the Help folder:

AX9LIB Reference Manual
AX9LIB Com Object Manual - UPDATE; Com Object has been replaced by .Net and Com Object will no longer be updated.
AX9LIB Logging
AX9LIBNET.CHM - AX9Lib is primarily a .NET object. Please review the .NET documentation for usage within a .Net environment.
AX9LIB Formats and Conversion - External V

Other beneficial reference material:

DSTU X9.37-2003 Specification for Electronic Exchange of Check and Image Data created by the Accredited Standards Committee X9, Inc.

American National Standard For Financial Services X9.100-180-2006 Specification for Electronic Exchange of Check and Image Data created by the Accredited Standards Committee X9, Inc.

American National Standard For Financial Services X9.100-181-2007 Specification for TIFF Image Format for Image Exchange created by the Accredited Standards Committee X9, Inc.

X9.37 Viewer is beneficial for using AX9Lib. A free download is available at www.allmypapers.com. To use X9.37 Viewer with full capabilities, a license is required. Please contact Sales@allmypapers.com.

Printing Document:

View>Normal (verify MarkUp is NOT set)
File>Normal Printing

Summary of Updates to Version 6.2.9 Manual August – 2014

The updates to the AX9Lib User Manual Version 6.2.9 thru July., 2014, – AX9LIB v6.2.9.

New functions:

New parameters:

AX9DemoNet - added return code entry field and renamed Return Code to Return Reason

New processing:

Ability to support/pass through Record 61 records for ConvertToUCD
Default Working Directory c:\Users\xxx\AppData\Roaming\AllMyPapers\Ax9Lib\work
ConvertTo180 - populate credit amounts and item counts in control records. Include credit images in the image count for control records
Record 28 - Set truncation indicator to "y" if there is no Record 26
Ability to support Source Rec 61 version = 58 for credit 25 for the first bundle. Add number of items and cashletter total to the deposit image.
For ConvertFromNSF, set the Bundle ID to DDHHMMSS
For Credit Record 28, set sequence number equal to credit 25 sequence number.
For Credit Record 26, set truncation indicator to "Y"
For Credit Record 28, set truncation indicator to "N"
Added Add26ForCredit and Add28ForCredit for ConvertFrom937 and Dest61>20
ConvertToNSFMit - Message in AX9LibNet that it is not currently implemented - obsolete
Handle Record 68 records for TestExchangeReady
Allow "" for data in SetAllFields and SetFields
For SetAllFields, if pass in null data then log, but don't return error
ax9lib.err moved to working directory
SetAllField - return error message if input data, record and field not set
Handle when X9ByteOrder and X9CharType set to unknown/not specified (introduced in 6.0.5)
Credit Records - Set routing number to the passed in DepositorRouting parameter
For credit items only, add a record 54 if there are already record 54s in the file
In Ax9DemoNet, added unknown option to Encoding, ByteORder, and TIFF Byte to dropdowns
Added help button (?) for Set Input Data and OutputData
Added default Depositor Routing and Account Number to blank instead of 0

Specialty Formats:

Support Canadian 015 format

Additional:

Improved logging
Allow ax9lib.ini to override the Enable Verbose logging and enable trace settings
Update AX9Libnet Help file
ConvertFromImages - NOT implemented.

Summary of Updates to Version 6.2.3 Manual Oct. – 2013

The updates to the AX9Lib User Manual Version 6.0.4 thru Oct., 2013, – AX9LIB v6.0.4.

New functions:

Addition of drop down boxes for:
ConverttoNSF; Sort: Forward to Return; Return to forward:Get Field Optimize
SortBOFDAccount – sort files by Record 26, Field 6
Sort Items
Convert to NSF/Mit
Micr Parse
Return to Forward – write a forward file from a return file
Write Return
Write Return Item

New parameters:

Addition of Enable Verbose Logging
Addition of Source Rec 68 Version parameter for supporting various Record68 input formats
Addition of Dest. Rec 68 Version parameter for supporting various Record 68 output formats
Endorsement parameters including Bank Name, Bank Routing, Deposit Account Number, Deposit Name

New processing:

Balance function modified to work with Record 61 type 47
NSF modifications:
NSF will read all 3 payee fields for record 26: payee, account, branch
Ability for NSF to input Deposit Branch number (R1,F5 - R10,F4 - R20,F4)
When Record 28generated via NSF, Endorsing Bank Date (F4) set to current date
Add NSF header record "28" to NSF file

CSV modifications:
Supports BOFA record 68 format. Generates a 81 character record.

Ability to support Record 68s for:
Wells Fargo, BOFA, and others
Set Record 52 sequence number to match record 25
Set Record 52 cycle number to match Bundle Header Record 20
Validate that Wells Record 68 are 80 bytes
Pass Source68 parameter into Print Ready and Exchange Ready

Added credit as a record 25:
Source61=38 creates R25 after each bundle
Set R52 sequence number to match Record 25
Set Record 52 Cycle Number to Bundle Record 20 Cycle Number

BOFA – Source 10/30/50/70 – Balance to not include R61s in Control counts.
Set Source61 parameter for Merge and Balance functions
Handle Bundle Limit
ConvertFrom937, set sequence number in Record 61 using the input data and Dest61=25/ 26
Dest=47 better processing for Record 26.

ConvertFrom937 will not change record 50, F3 if it is present in the source file.

Endorsement:

Endorsement Option requires license and Auto Deposit
Added Record 68 support to endorse
Ability to use Destination file for Endorsements
When Record 28generated via NSF, Endorsing Bank Date (F4) set to current date

Handle all BOFA formats in Balancing

Ability to set sequence number in Record 61 with Dest61 = 25 or 26 and using input data
With ConvertFrom937

Ability to use Destination file for Write Return processing

Sort:

SortBOFDAccount – ability to sort files by Record 26, Field 6
Sort Items – Optional fields are Destination File (defaults to Working Directory), Record 25 or 31 (defaults to 25 if not specified and Field (7 or 8, defaults to 8 if not specified – sequence number.)

Deposit Slip:

Auto creation of Deposit (Need endorsemement bit)

Balance and Test Balance:

Modified to work with R61 following a Cash Letter
Routines using did not distinguish between blank and zero in R70, F4.
The effect was a delete in some cases would change the MICR Valid Amount.

Specialty Formats:

Record 61 Formats – AX9lib will create record

BOFA – Dest Rec. 61 Version=50

Wells Fargo - Dest Rec. 61 Version=45 or 25(enter Seq# and Account#)
When updating R61, F6 (Wells) use SourceVer=45 using the SetInputData field to update Serial #
Dest 61=25 or Dest=26, set sequence number using the input data

X9.100-180 - Dest Rec. 61 Version = 33 (84 bytes)
Dest Rec. 61 Version = 47 - better handling of R26 - needed editing

Modified Set All Fields, Set Field, and Get Field when updating a Record 61using Source Dest = 5

Added better handling for record 61s for balancing files and cash letters

Credit Record 25 Format

Dest Rec. 61 Version=38 (38,58,78)
Auto generates deposit tickets after each bundle record
Requires Endorse Edit bit for Auto Deposit

Wells Fargo - ConvertFrom937 pass in Serial # for R61,F6 in Data field (Program left justifies)

Record 68 Formats – 68 record must be included in incoming file

Version Source 68 support added for different versions

BOFA	Dest Rec. 68 Version=10 (81 bytes) (No R61) CSV will support BOFA record 68 format. Generates an 81 char record.
Wells Fargo	Dest Rec. 68 Version=5 (936 to 937)
Wells Fargo	Dest Rec. 68 Version=6 (937 to 936)
Special Format	Dest Rec. 68 Version=14 (200 bytes)

Additional:

AX9LibNet released

MICRPARSE added to AX9LIBNet

MicrParse - default chartranslator to "0123456789ABCD *N" if not passed

EOFB test can be turned off by user.

ConvertFrom937 will support WELLS no header format with a non compliant Record 68.

Sort/Detect dups and Sort Dups (No Amount) writes out .1st and .937 files

Documented Sort, SortPayor, Sortbofd, Sortbofdaccount

Standard level 3 and SVCPCO 13 compatible and standard level 30 for 187-2013

Summary of Updates to Version 2.4.5 Manual May - 2011

The updates to the AX9Lib User Manual for May, 2011, Version 2.4.5.

New functions:

Get License Availability displays license availability

New parameters:

Addition of optional Record 28 to NSF/NSF2 format
Addition of 18700 for UCD records using Get and Set

New processing:

Sequence numbers are now auto incremented when doing a record 61 insert (various subformats).

.

Set Field can access records with 187 format by adding 18700 to the record number.
Get Field can access records with 187 format by adding 18700 to the record number.
An Endorsement Record 28 will be added with an inclusion of a "28" in the NSF/NSF2 file.

Additional:

Updated Error List

Summary of Updates to Version 2.4.4 Manual August - 2010

The updates to the AX9Lib User Manual for September, 2010, Version 2.4.4 include a new Test Progress Bar.

New functions:

Test Progress Bar
Test Progress Bar Ex

New parameters:

New processing:

Test Progress Bar buttons shows processing progress
Convert toUCD converts nulls to blanks, justifies return record/fields 31,4 and 33,3, and forces record 31,8 to be the same as 10,9.
Sort/Detect Dups now generates two picklists of duplicates; source and destination.
SortPickList property used in Sort/Detect Dups to point at duplicates in the source X9.37 file.

Summary of Updates to Version 2.4.3 Manual – March 2010

The updates to the AX9Lib User Manual for March, 2010, Version 2.4.3 addresses several new functions pertaining to sorting duplicates.

It also adds a screen function for converting a 937 format to a UCD 187format. (A 180 license is necessary.) This function will eliminate the need to place a 15 in the Dest. Rec 61 Version field.

New functions:

Sort Payor
Group Account
Get Items in File
Get Header in File

New parameters: Endorsement Options

Sort Dups Start Item

Summary of Updates to Version 2.4.2 Manual – January 2010

The updates to the AX9Lib User Manual for January, 2010, Version 2.4.2 addresses several new functions pertaining to sorting duplicates.

It also adds a screen function for converting a 937 format to a UCD 187format. (A 180 license is necessary.) This function will eliminate the need to place a 15 in the Dest. Rec 61 Version field.

New functions:

Convert to UCD
Sort/Detect Dups
Sort Dups (No Amount)
Sort Micr Fields

New parameters: Endorsement Options

Deposit/Payee Endorse
Bank/Transit Endorse
BOFD Endorse
Horizontal Box Endorse
Vertical Box Endorse

Summary of Updates to Version 2.4.1 Manual – November 2009

The updates to the AX9Lib User Manual for November, 2009, Version 2.4.1, addresses new processing for creating Endorsements on the back of a check image. This is done by drawing an endorsement on the back of every check in the file. This update also supports the creation of a UCD 187 compatible format from a X9.37 file. (A 180 license is necessary.)

New function:

Endorse 937

New parameters: Endorsement Options

Bank Name
Bank Routing Number
Deposit Acct Number
Deposit Name
Font Style
Drawing Options
Endorse Height (in)
Endorse Width (in)
Bottom Offset (in)
Right Offset (in)

New processing:

Ability to draw an endorsement on the back of the checks.
Ability to support an UCD compatible output file from a 937 file.

Other Modifications:

None

Summary of Updates to Version 2.4 Manual – September 2009

The updates to the AX9Lib User Manual for September, 2009, Version 2.4, address new processing for creating Returns from an entire input file and creating/writing MAL files.

New functions:

Convert to MAL - Read X937 and convert to Fed Payer Services ASCII file of 937 data with non-standard header format.
Convert From MAL - Read Fed Payer Services ASCII fil (MAL0 and write X937
Write Return All

New parameters:

None

New processing:

Ability to write a return file from an entire input file
Ability to create a .pic list of items not containing a Record 26
All return generation functions now generate a Destination and/or Pick file if parameter present
When there is no Record 26, the nOccurence parameter when set to 99 will use the most recent Record 28 as the return routing number. If any other value, the the first record 28 will be used
Additional support for SOP 4.8 files from different districts

Other Modifications:

Subformat 47 has an improved record 50, field 7
If producing record 61 with Regions bank sub-format, a 54 is not included in the image set
All set field of record 61 for subformat versions 1,2,3 defined by source record 61

Summary of Updates to Version 2.3 Manual – May 2009

The updates to the AX9Lib User Manual for April, 2009, Version 2.3, address new processing for creating Forward items for Returns .

New functions:

Convert to SOP
Merge Files (replaces Merge List)
Write Return Item – create a Forward Representment from Return item
Write Return – create Forward Representments from a list of Returns

New parameters:

None

New processing:

Ability to merge files into 1 file just as they are
Ability to create Forward items from Return items
Ability to create Regions format Record 61, 84 bytes, no Record 26
Ability to create Wachovia (additional format) Record 61, no Record 26

Other Modifications:

Screen – VBAX9Demo screen has been replaced with cppAX9 Demo
All references to VBAX9Demo has have been replaced to the cppAX9 Demo
Set All Fields updated for 84 byte variation of record 61
Single Item returns have return reason in Record 35
Cash Letter ID (Record 10, field 10) carries value from file on ConvertFrom937

Summary of Updates to Version 2.2 Manual – October 2007

The updates to the AX9Lib User Manual for October, 2007, Version 2.2, address new processing for Returns and not correctable items.

New functions:

SortBoFD
Not Correctable List

New parameters:

None

New processing:

Ability to sort Return files by Bank of First Deposit (BOFD). Uses the auxiliary file as a translation table to convert BOFD to Return Processor Routing Number. This is a file created by the institution using the product.

Return output format now matches input if no conversion is specified. Previously the format was fixed to EBCDIC/Big Endian.

Ability to create a subset list with items that are not correctable in the Convert From 937 function. It will read a pick list file, determine which items are correctable to the X180 TIFF standard, and then write out a new pick file which lists just the ***not*** correctable items. The output list can then be used along with the List Delete function to edit out those items from the X9.37 file prior to using Convert From 937 to correct the non-conforming images.

Other Modifications:

Screen – AX9Tools screen has been replaced with VBAX9Demo

All references to AX9Tools have been moved to the AX9Tools document.
Get License now available through Get Version

Summary of Updates to Version 2.1 Manual – June 2007

The updates to the AX9Lib User Manual for June, 2007, Version 2.1, address maintenance upgrades and modified processing.

New parameters:

Set Input Data:

Delete record types indicated in the field that are contained in a X9.37 file using the Convert From 9.37 function. Specific Record Types are allowed.

Set Input Data:

Input parameters will be placed in this field for the Set and Set All functions, instead of the Return Code/Data field.

New/modified processing:

Enhanced ConvertFrom937 functionality

Ability to delete records contained in Set Input Data parameter

If resolution set not at 200, but data appears to be 200, set resolution to 200.

Enhanced ConvertFromCSV

Uses Record 61

Enhanced Print Ready and Exchange Ready tests

Generates Output Pick List when items fail PrintReady or Exchange Ready test

Enhanced Get/Set functionality

Supports records 50, 52, and 54

Enhanced Record 61 functionality

Support for additional credit records. (Contact AMP for specifics.)

Summary of Major Updates to Version 2.0 Manual – Dec. 2006

The AX9LIB SDK was originally developed to support ANS X9.37 DSTU. On July 12, 2006, the X9 Committee approved the X9.100-180-2006 standards which replaces the DSTU X9.37.

The actual usage of X9.37 has created numerous variations (proprietary versions)of the file format. The new version of AX9LIB supports new functions to deal with the conversion and usage of these non standard formats of X9.37 that have evolved, as well as the new X9.100-180 files. A new function call will allow you to generate various kinds of X9.100-180 files based on the variety of X9.37 files you currently have.

AX9Lib has also morphed into AX9Tools. This tool allows all the functionality of AX9Lib and allows the functions to be run in script/batch file mode using Command Line Processing. All the screen parameters can be loaded into an .INI file using the GUI for AX9Tools, then saved and reused as needed. A series of scripts can be run to accomplish the major AX9 processing functions.

The updates to the User Manual for December, Version 2.0, address the new functions and parameters. See Input Screen Definitions for a description and the use of the new functions and parameters.

New functions:

Merge Cash Letters
Merge Bundles
Merge List
Limit Bundles
Convert to 180

New parameters:

Destination File
Report File
Pick File
Cash Letter ID
Cores
Bundle Limit
Repair Threshold
Repair Max Image Size
Repair Images

New/modified processing:

Record 61 – creation, conversion, insertion and balancing of files with many different Record

61formats. (Note: not all functions support Record 61.)
Creation of Record 61 into an existing X9.37 file on either a bundle or cash letter basis.
Support of Record 61 (Version 1) for 180 format.

File reports/logs – XML "payload" File with a summary of the file contents
Log File – resides in the Working Directory and includes
enhanced detail information on each item that fails compliance
Error Pick File – resides in the Pick File field if specified

Semaphore file – a "destination" file is supported for many functions. When the parameter is
present, the resulting file is moved to this location and a semaphore file with the same name with an ".sem" extension is created in the destination directory. By sensing the semaphore file, another process can determine that the destination file is available for additional processing.

File processing – decreased processing times on large files using multiple cores.

X9.37 File Conversion – conversion of existing X9.37 files to universally compatible X9.37 including TIFF image conversion, Record 61 conversion and file format conversion
(ASCII/EBCDIC, Motorola/Intel)

Image Repair – repair images too large to be accepted for exchange. Removes background noise and actually improves the readability of the image.

File Testing – detects format errors
Test file for IRD Print compliance
Test file for Exchange compliance

Compliance Failure - supports generating a "Pick File" of items that fail either the Print compliance or the Exchange compliance testing. This "pick file" can be used to delete or copy items from the source in a second step.

Other Modifications:

Screen – AX9Demo screen has been replaced with AX9Tools

Name Modifications:
File Name to Source File
Aux. File Name to Auxiliary File

New Report Example Section
XML Report – Report File field if specified
Log File Report – Working Directory
Error Report – Pick File field if specified

Working Directory

The user may specify a Working Directory to hold temporary files created by AX9Lib during X9.37 file processing.

The files placed in the Working Directory will *normally* contain **output** files. You can create this directory within AX9Lib by double-clicking Working Directory, determine the drive where it is to reside, and click Make New Folder at bottom of screen. It is recommended you create this temporary file prior to executing any AX9LIB functions.

Warning!
Under no circumstances should the Working Directory be the same as the directory containing the input source X9.37 or the CSV file.

When creating a CSV file from a X9.37 file, the variable length files, especially the images, will be placed into the Working Directory. DO NOT delete the Working Directory until you are done with the CSV file.

Input Files

The input file name (*normally* a read only file) is placed in the File Name field on the screen. The exception to this is the Append/Prepend Functions where the Source File field becomes an output file (write) and contains the name of the file to which you want the records appended/prepended.

The Auxiliary File field on the screen is also an input field and contains additional information for various functions of AX9Lib.

Output Files

The Working Directory will normally contain the output files of the function and will be named by the system. They are usually created using the File Name name and appending the type of file and function you are creating. Many of the files can be opened using NotePad and others AMP X9Viewer. Some are AX9LIB processing logs.

Example:	Function	File extension
	Convert Functions –	CSV, 937, MAL, NSF (type of file produced)
	Copy/Delete Functions -	_Copy.937
	Balance -	TOC.SUM (Table of Contents)
	Split Bundle -	_BUN_1_1.937
	Split Cash Letter -	_ICL_1.937
	Write Return -	_COPY.R37
	Print/Exchange Ready-	ANA (Classification Frequency Report on Images
	Sort/Merge -	AMP.TOF (A list of files created by function)
	System logging	Text - Runtime Log (called AX9LIB)
		ERR - Processing Error Log
	Additional logging file extensions:	
		ANA, TOC, SEM, LST, TOF, TOC, TEXT,

The Destination File and Report File fields designate a specific file name for the output instead of letting it default to the Working Directory with the File Name name and extension.

Pick File can be either an input or output file depending upon its use. It is used as an output file when you use the Test Exchange Ready and Test Print Ready functions and an input file when using the Find Pick 937 function.

ASCII File – Selection Lists

AX9Lib can support two different "selection list" types which are used for various screen functions. Usually, the Auxiliary File field holds the pathname for the Selection List file. In some instances, it may be useful to set Auxiliary File to NULL and let AX9Lib look for the list file in the Working Directory.

Selection Lists are ASCII formatted text files (or parameter lists) containing either:

- A list of *values(Find List) or*
- A list containing the Cash Letter, Bundle, and Item numbers (Pick List). This list is separated by commas.

Normally the set of functions that use these lists is concerned with selecting the records for additional processing.

Find List – a list of values from a file (not just a cash letter)

This type of list is used in the functions:

Find Pick X9.37
Select List 9.37
Select List CSV

A typical example is the Item Sequence Number. If the value is unique such as an Item Sequence number, then only one item will be found. If a value is a Routing number, then all occurrences of the Routing number in the record/field that is specified will be found.

The Find List itself only contains the *value* to be found. Since this is a direct pattern match, the value should be in quotes if blanks are included.

The screen parameters, Record and Field, define the record type and field number that contain the data that is to be searched. Only one record/field combination can be used per Find List as these are entered through the screen and not included in the list.

Example: Find all occurrences of routing number 011 00011, 123456789, 987654320:

ASCII "Find List" file would contain:
"011 00011 " (use the quotes when a space is included in the field - do not need quotes if spaces at the beginning or end)
123456789
987654320

Screen Field parameters would be set to:
Record: 25 (this is Item record)
Field: 4 (this is the Payor Bank Routing Number field)

AX9Lib function Find Pick 9.37 will read the "find list" records and read the input file and will only write the cash letter number, bundle number, and item number of the records in the output file that fit the criteria specified by the "find List" file and the screen parameters. In the above scenario, this would include all the records containing the routing numbers above in record 25, field 4.

As stated, the *output* record of this function will only contain the Cash Letter, Bundle, and Item Number of the selected records. The triplet can then be used in the Pick List functions.

Pick List – Since X9.37 files can be large, it is often efficient to create a Pick List of all the items in the file that need some form of processing.

A "Pick List" is just an ASCII text file containing Cash Letter Number, Bundle Number, Item Number and in the case of returns, the fourth field can contain a Return Code. The values are separated by commas. These are used to pick items from a cash letter. **This is used in ICL processing, not file processing.**

This type of list is used in the functions:

List Copy
Append 9.37
Append CSV
Prepend 9.37
Prepend CSV
List Delete
Write Return

Example of Pick List text file:

1,2,3
1,2,4

These two lines of text will select the 3rd and 4th item from Cash Letter 1 and Bundle 2 and execute the function selected using these records.

List Versus Item

The same screen Function is often repeated with either List or Item in the name. E.g., Select Item, Select List, Write Item, Write List.

A List is a *file* and usually contains *multiple* items to process.

An individual X9.37 *item* consisting of record types 25-54 or 31-54 may be selected by entering the three screen parameters: Cash Letter Number, Bundle Number, and Item Number.

Append/Prepend - Examples of Insertion

When building a new file with additional items, it is sometimes convenient to simply append or prepend to the end or the beginning of the file.

The Append and Prepend functions will insert items from the Auxiliary File into the File Name based on the three parameters of Cash Letter, Bundle and Item. For the Append/Prepend functions, these three screen parameters identify the placement of the new records in the existing file.

Note: For these functions, it is a good idea to save a copy of the original file, because AX9Lib actually appends/prepends to the original File Name. It does not create another file in the Working Directory.

The following table provides examples of such effects.

Location	Prepend	Append
0,0,0	Before first item of first bundle of first cash letter	After last item of last bundle of last cash letter
N,0,0	Before first item of first bundle of Nth cash letter	After last item of last bundle of Nth cash letter
N,M,0	Before first item of Mth bundle of Nth cash letter	After last item of Mth bundle of Nth cash letter
N,M,P	Before Pth item of Mth bundle of Nth cash letter	After Pth item of Mth bundle of Nth cash letter

X9.37 Record Types

(Viewed by using X9Viewer)

Header

01--File Header Record

10--Cash Letter Header Record

20--Bundle Header Record

Check Detail

25--Check Detail Record

26--Check Detail Addendum A Record

27--Check Detail Addendum B Record

28--Check Detail Addendum C Record

Return

31--Return Record

32--Return Addendum A Record

33--Return Addendum B Record

34--Return Addendum C Record

35--Return Addendum D Record

Totals Detail

40--Account Totals Detail Record

41--Non-Hit Totals Detail Record

Image View

50--Image View Detail Record

52--Image View Data Record

54--Image View Analysis Record

Credit Detail

61--Credit Detail Record

Bundle Control

70--Bundle Control Record

Summary

75--Box Summary Record

85--Routing Number Summary Record

Control

90--Cash Letter Control Record

99--File Control Record

Item Detail Record Content

The MICR data in a X9.37 file record 25 (Check Detail Record) is parsed data. This means that the MICR line data is represented by multiple fields and the ***creator*** of the file must fill the fields appropriately. The method used to populate the MICR data fields will vary based on the source of the MICR data.

The MICR fields used in X9.37 going from left to right on a check are:

Auxiliary On Us
External Processing Code
Routing Number
On Us
Amount

NOTE: The X9.37 file does not have a specific field for either check number or transaction code.

Deposit Slips

The X9.37 DSTU does not have a specific record for Deposit Slips (Credits). Because of this, multiple methods are in use to convey this information. The appropriate format is, again, a function of the Companion Document for the *receiving* institution. In general, to create a file that can be used with multiple receiving institutions, multiple methods should be selectable. Of course producing no deposit slip record at all is required by major exchanges including the FRB.

There are three common methods of entering deposit slips and of course variations on each.

Non valid ABA number : A valid (R25,F4)ABA routing number is identified by the first 2 digits of the routing number which ABA has reserved to identify the Federal Reserve Districts and the variations on this format. (See next page.) When an item has a non-valid ABA number, then it can be assumed to be a deposit slip since a check will have a valid routing number.

Record 61: The newest generation of X9.37 will have a record 61 which is a specific Credit Record 61. Unfortunately there are multiple versions of this record in common use. **To provide a file with a specific Record 61, the Companion Document from the receiving bank will identify the correct format.**

Of course, it is also possible to build one format and use the AX9Lib tools for conversion to the others.

A Record 61 is a full "item" of its own. It will typically have image records for the front and back. They generally do not have other addendum records, but various companion documents do specify addendums.

Trans Code on a valid ABA number: A normal ABA routing number with a trans

code to identify it as a Deposit Slip will generally print an IRD as a "Legal Copy" unless special software is written to identify the Routing number/trans code combination. See your vendor.

Valid ABA Numbers

The first two digits of the US routing numbers have been assigned by the ABA to Federal Reserve Districts and to special variations. The valid numbers are listed.

First two digits		
00	Federal govennment	
01-12	FRB District	
21-32	Thrift in District	
61-72	Electronic	(should never appear)
80	Travelers Check	

Control Records

The control records provide an audit for the contents of the file. There are control records for the file (record 99), the cash letters (record 90) and the bundles (record 70).

In general the control records will sum up to the number of items and the total amount in the appropriate record. This is all obvious and well defined until deposit slips are added. Since they were not accounted for in the X9.37 DSTU, they will be handled differently for each type of Deposit Slip and for certain Record 61 formats.

Non valid ABA number: The deposit slip will be a standard record 25 item. This means that the amount of the deposit will be included in the total amounts in the control records. In this case the totals will all be double the actual check contents.

Record 61: This is known to be a Credit record and its **amount** is ***NOT*** included in the total

amount for control records**.** However, Record 61 will be counted in the **total items** because it is an item.

A variation of this is the DestRec61version = 50 format(BOFA), where balancing will **not** include the totaling of the Record 61 for either the amount or the count.

Valid ABA number with a special trans code: these deposits will be processed just as a standard record 25. It will sum to double the actual check amount in the file.

Date Format

All dates should be entered as yyyymmdd (20140624)

AX9DEMO - Dashboard

Note: The bottom 1/5 of the screen varies depending on if additional parameters are needed to complete various functions, e.g Endorsement parameters, Destination Credit parameter, or Sort parameters.

For each AX9LIB Function, the full pathname of the primary File Name must be entered.

It may be necessary to enter other screen parameters that apply to the function being requested.

User entry fields (Parameters) are on the left-hand side of the screen and the Functions that can be performed are in the columns on the right-hand side.

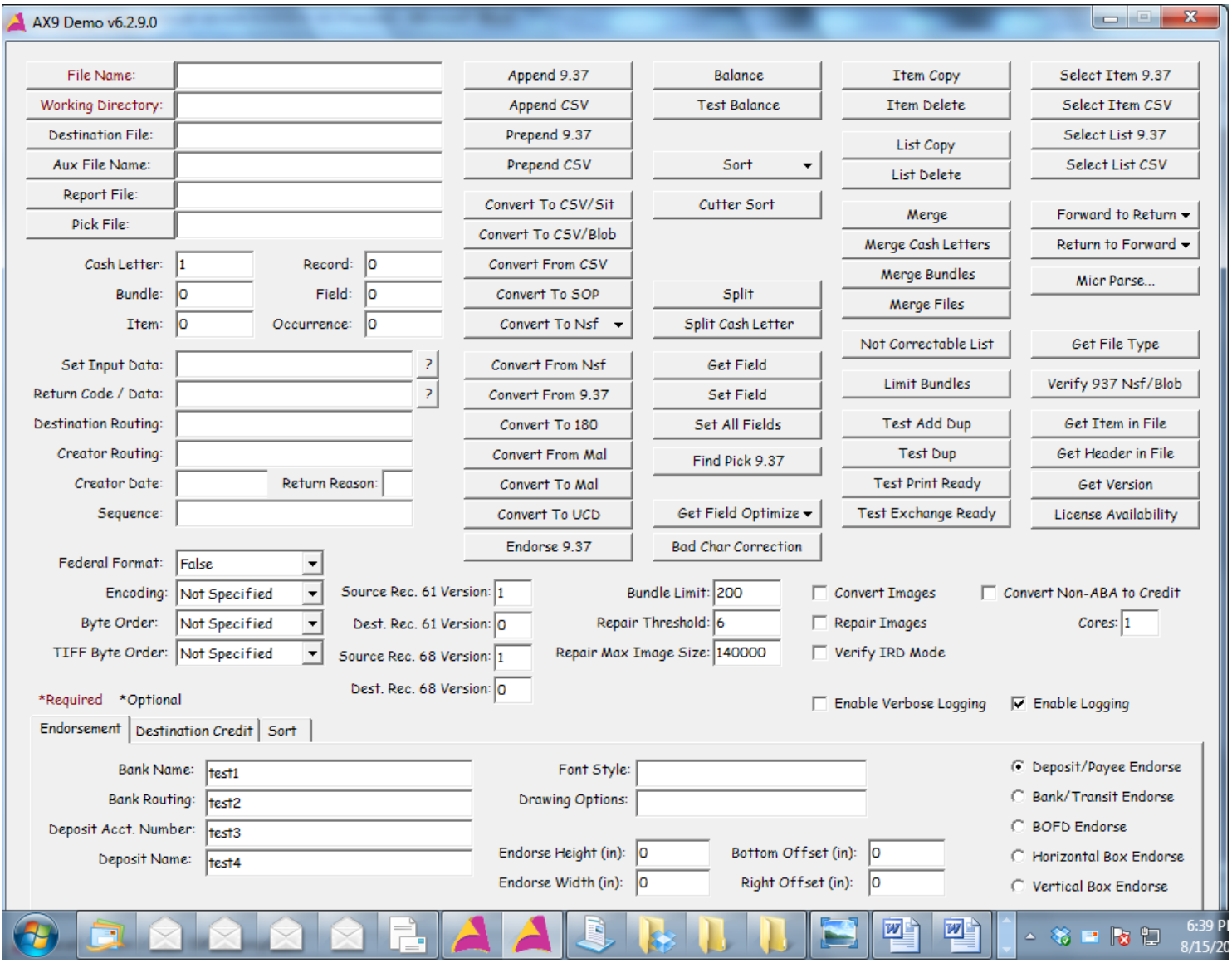
The following pages give example of each Function and the Parameters that are needed. Test default values display in this example. All successful functions will display a message on the screen, i.e.,

Convert From NSF
Completed Successful

If processing errors occur, they will also be displayed on the screen with an error number, i.e.,

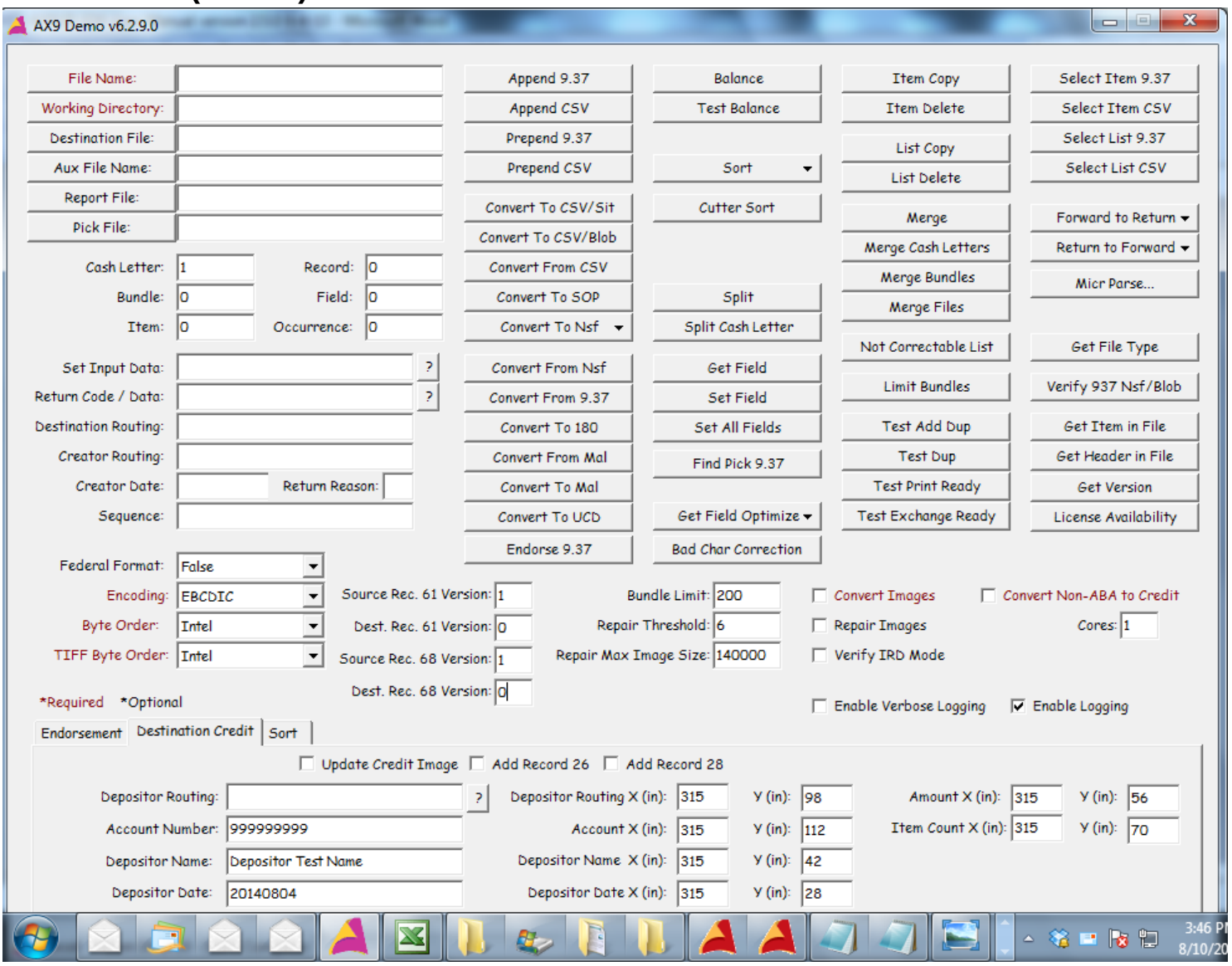
Convert From NSF
Returned with error = 96 (See AX9 Error Codes in Appendix A.)

Endorsement Screen Parameters



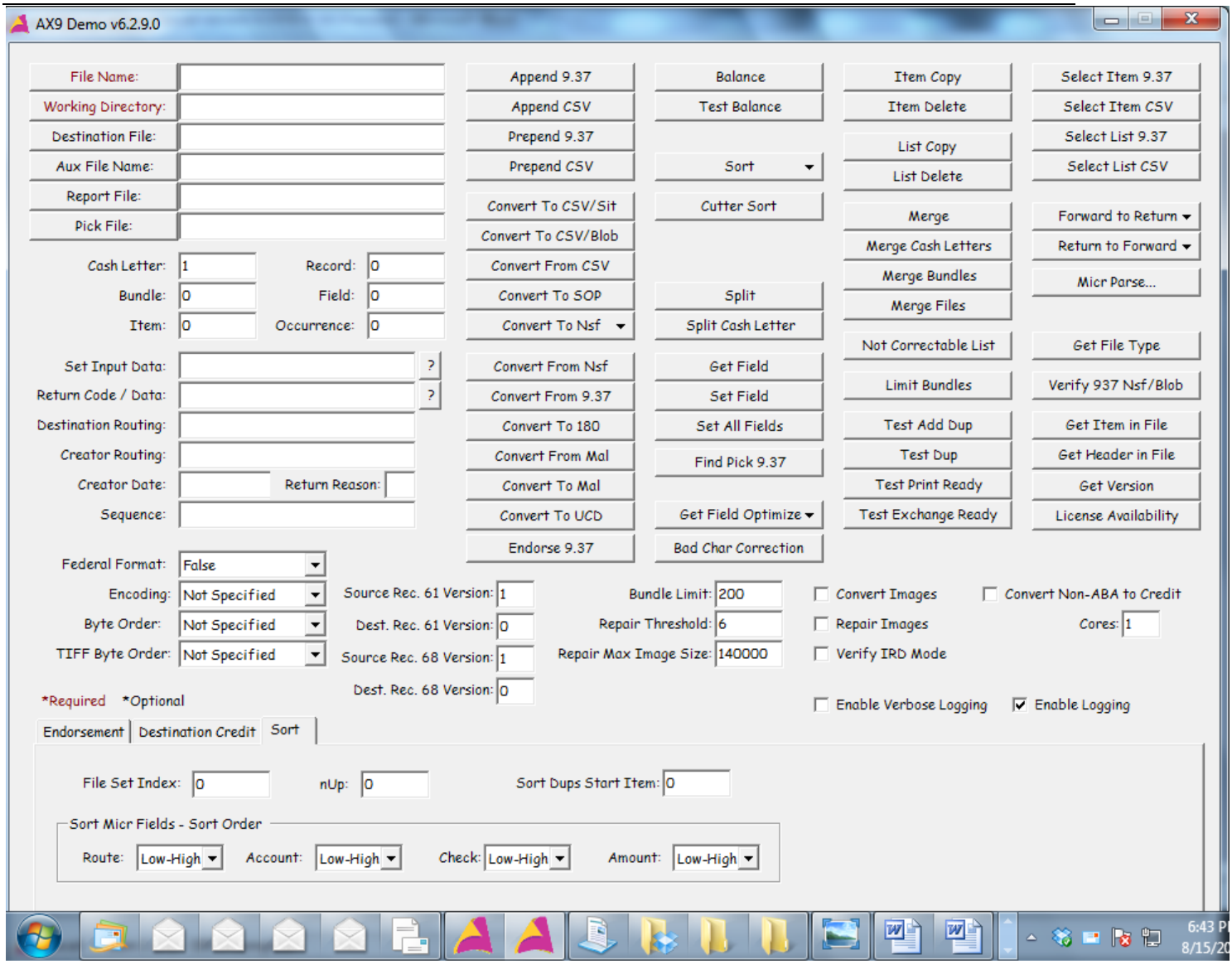
Endorsement fields - bottom of screen starting with Bank Name

Destination (Credit)Screen Parameters



Destination Credit fields- bottom of screen starting with Depositor Routing:

Sort Screen parameters



Sort fields- bottom of screen starting with File Set Index.

Summary of AX9 Screen Functions

(The support for Records 61 and 68 is ongoing and may not be supported by all functions.

Read/Write Functions

Convert From 9.37 – Read X9.37 and Write X9.37

Convert From CSV – Read CSV and Write X9.37
Convert to SOP – Read X9.37 and Write SOP

Convert to CSV/Sit – Read X9.37 and Write CSV and TIFF

Convert to CSV/Blob – Read X9.37 and Write CSV

Convert to Nsf:

Convert to Nsf/Sit - Read X9.37 and Write NSF and TIFF

ConvertToNsf/MIT - Not implemented

Convert to Nsf/Blob - Read X9.37 and Write NSF

Convert From Nsf - Read NSF and Write X9.37

Convert to 180 – Read X9.37 and Write 180
Convert to MAL - Read X9.37 and Write MAL
Convert From MAL - Read MAL and Write X9.37
Convert to UCD (187) - Converts a standard X9.37 file to a UCD file
Endorse 9.37 - Adds endorsement to the rear of check images

Editing Functions

List Copy – Copy a List of items to a new file

List Delete – Delete a list of items from an existing file

Item Copy - Copy a single Item to a new file

Item Delete – Delete a single item from an existing file

Split Functions

Split – Extract *Bundles* into multiple files

Split Cash Letter – Extract *Cash Letters* into multiple files

Append/Prepend Functions

Append 9.37 - Append X9.37 items to a X9.37 file

Append CSV – Append CSV items to a X9.37 file

Prepend 9.37 - Prepend X9.37 items to a X9.37 file

Prepend Csv - Prepend CSV items to a X9.37 file

Balance Functions

Test Balance – Verify balances in the file

Balance – Balance File

Field Functions

Get Field – Retrieve data from fields of X9.37 file

Get Field Optimize - Retrieves data from fields of an X9.37 file using an already TOC file. (Specific steps.)

Set Field - Place data in a specific field in X9.37 file

Set All Fields - Set all occurrences of a field in X9.37 file

Select Functions

Select Item 9.37 – Extract a single item from a X9.37 file

Select List 9.37 – Extract a list of items from an X9.37 file

Select Item CSV – Extract a single item from an X9.37 and write CSV

Select List CSV – Extract a list of items from an X9.37 file and write CSV

Find Pick 9.37 – Extracts a list of items designated by an input text file from an X9.37 file and writes a text file containing the results

Sort/Merge Functions

SORT:

Sort – Extract items based on Payor Bank routing number (Forward processing)

Sort Items – Sorts file by record 25 or by record 31 for numeric fields only.

Sort BofD – Used in conjunction with Merge to group items by BoFD Routing (Record 26, Field 3). Creates an intermediate file for each Bofd Routing number in the working directory. Also creates AMP.TOF containing a list of these intermediate files .

Sort BOFD Account - Used in conjunction with Merge to group items by BoFD Account (Record 26, Field 6). Creates an intermediate file for each Bofd Routing number in the working directory. Also creates AMP.TOF containing a list of these intermediate files.

Sort Payor - Same as Sort except can pass in Auxiliary file. Used in conjunction with Merge to group items by Payor Routing (Record 25, Field 4). Creates an intermediate file for each BoFD Routing

number in the working directory. Also creates AMP.TOF containing a list of these intermediate files

Group Account - will extract items based on ON-Us field (R25,F6)

Sort/Detect Dups - The SortDetectDups method performs a sort operation on an X9.37 file to identify duplicate items in the file. It is a quick way to determine if a check has been submitted multiple times

Sort Dups (No Amount) . SortDetectDups with Amount disabled, will perform a sort operation on an X9.37 file and identify duplicate items in the file, but will not use the Amount field as criteria, only the sorts by Payor Routing, Account Number and Check Number. When used with a Merge command, this can be used to perform duplicate detection on a multiple day range of input.

Sort MICR fields - Extracts items based on MICR fields

Cutter Sort – Sorts the records of a 937 file. This method is used to sort X9.37 files in preparation for IRD Printing. The fileName is sorted so that the output can placed on a multiple page cutter and all the pages cut at once. In order that the resulting stacks of IRDs can be bunched together, multiple pages will have VOID checks inserted as needed

Other Merge Functions

Merge – Combine a list of files

Merge Cash Letters – Combine multiple cash letters within a file into a single CL

Merge Bundles– Combine all bundles within a cash letter into one bundle per CL

Merge Files – Combine all files contained in a list into separate CL in one file

Limit Bundles - Create bundles with a set number of items

Return Functions

Forward to Return:

Write Return All - Create Returns from entire input file

Write Return – Create a Return from a list

Write Return Item – Create a Return from a single item

Write Return Find List – Create a Return from a search list

Returns to Forwards:

Write Return – Create a Forward File from a Return list

Write Return Item – Create a Forward File from a single item

Duplicate File Test Functions

Test Add Dup – Add entries to a History file

Test Dup – Searches a History file for duplicates

Miscellaneous Functions

MICR Parse -Parses a MICR line from a string

Get File Type – Retrieve file format information

Get Version – Retrieve version of AX9Lib

Get Items in File - displays item count of Record 99

Get Header in File - displays string contents of Record 01

License Availability - Indicates if a license is available or not

Test Print Ready – Tests X9.37 file is ready to print IRDs - Annex F compliant

Test Exchange Ready - Verifies file is properly formed for general X9.37 exchange

Verify 937 Nsf/Blob - Verifies the MICR values of the front check images with the MICR values contained in the X9.37 file and output a NSF/BLOB file. Requires MicrBatch.exe, MicrBatch64.exe and MicrBatch.ini

Not Correctable List - Creates a new picklist file that only has the 937 items that are not correctible in a ConvertFrom937 operation

Endorse 937 - Adds an endorsement to the rear check images in the X.937 file

Test Progress Bar - API Disabled

Bad Char Correction - Tests for bad characters and creates a report. If a destination file is specified, corrects bad characters and generates a new file with the corrected characters.

AX9Lib – AX9DEMO Screen Definitions (Input Fields).

Many of the fields on the screen are optional and depend on the Function that is used. The parameter input fields (all input fields except for the file fields) will default to the values in the file if they are needed by the Function requested and are not entered

Field Name	Description, Valid Codes, and Comments	Function Use
File Name	Full pathname to input source file	All
Working Directory	Full pathname to *directory* where the output files will be placed. (normally).	All
Destination File	Allows use of a specified output file Full pathname – system will generate a 937 file with the items	Merge Cash Letters Merge Bundles Merge Files Limit Bundles
	A 937 file will be created with items that fail to verify	Test Print Ready Test Exchange Ready
Auxiliary File Name	Input containing a parameter list or file. Full pathname to 9.37 records to be inserted Full pathname to CSV records to be inserted	Append937 Prepend937 Append CSV Prepend CSV
	Full pathname to text file	Copy List Delete List
	Full pathname to text file	Select List 937 Select List CSV
	Full pathname to text file of Return Processors For a given BOFD	Sort BoFD
	Full pathname to text file	Find Pick 937
	Full pathname to text file	Write Return Find List Write Return
Report File	Output - Allows use of a specified XML report file Created from the destination 937 An XML individual item Error Report for Test Ready Exchange	Test Print Ready Test Exchange Ready Test Balance Convert From 937 Convert From NSF
Pick File	Output containing items that fail to verify Full pathname Output - Full pathname to text file Output - Full pathname to text file	Test Print Ready Test Exchange Ready Find Pick 937 ConvertFrom937

Field Name	Description, Valid Codes, and Comments	Function Use
Pick File	Input - List of items with no 26 record	(Non correctable list) Write Return All
Cash Letter	Specifies which existing cash letter is being addressed Valid Values: 1-999999999999	Item Copy Item Delete Select Item 937 Select Item CSV Get Field Set Field Write Return Item
	Specifies where the items are to be inserted	Append./Prepend 937 Append/Prepend CSV Append/Prepend CSV
Bundle Item Copy	Specifies which bundle within the cash letter is being addressed. Valid Values: 1-999999999999	Item Delete Select Item 937 Select Item CSV Get Field Set Field Write Return Item
	Specifies where the items are to be inserted	Append./Prepend 937 Append/Prepend CSV
Item	Specifies which item within the cash letter is being addressed. Valid Values: 1-999999999999	Item Copy Item Delete Select Item 937 Select Item CSV Get Field Set Field Write Return Item
	Specifies where the items are to be inserted	Append./Prepend 937 Append/Prepend CSV
Record	Specifies which existing record is being addressed is being addressed. Valid Values: 0-999999999999 Valid Values: 01, 20, 20, 25, 31, 50, 52, 54, 70, 90, 99	Item Copy Item Delete Set Field Set All Get Field

Field Name	Description, Valid Codes, and Comments	Function Use
Field	Specifies which existing field is being addressed.	Item Copy Item Delete
	Valid Values: 0-999999999999	Select Item 937 Select Item CSV Get Field Set Field Set All
Occurrence	Specify when there are multiple identical record numbers in an item (e.g. endorsements), set the Occurrence value to the desired record. Use an Occurrence value of 0 to get a particular field from the first of multiple X9.37 records. In a similar way use an Occurrence value of 100 to point at the last record. Valid Values: 0-999999999999	Get Field Set Field
Set Input Data 9.37	Delete all record types indicated in this field from the file Values: 10, 20**,** 26, 27, 28, 33, 34, 35, 50, 52, 54, 70, 90 Not Recommended Values: Deleting the **first** record of an item could be a catastrophe if using for a production file, e.g. 25, 31, and 61. If deleting a record for testing, most can be deleted to create an invalid file. When using this function ***all*** records indicated are deleted from the file.	Convert From
	Input data – set the value of the data you want entered into the record(s) and populate the CL, Bundle, Item, Record, Field parameters with the location of the record you want changed. If you want all the records in the file changed to this value it is not necessary to populate, use Set All.	Set Field Set All
Return Code/Data	A code that indicates the reason for non payment Valid Values: *See section 14.6 DSTU X9.37-2003 specifications For comprehensive list of Return Reason Codes*	Write Return Item
	Used in the Test Dup Processing functions to indicate the duplicate file found in the history file if the error return = 143 (File Duplicate)	Test Add Dup Test Dup

	Used to display the value of the data	Get Field

Field Name	Description, Valid Codes, and Comments	Function Use
Destination Routing	Used in the Return Processing functions (Record 10) Valid Values: 0-999999999	Convert From Nsf Set Field Convert From 937
Creator Routing	Used in the Return Processing functions (Record 10) Valid Values: 0-999999999	Convert From Nsf Write Return Write Return Item Write Return Find List Convert From 937
Creator Date	Date used in the Return Processing functions (Record 10) Valid Values: yyyymmdd Find List	Write Return Write Return Item Write Return
Sequence	The numeric value of the Sequence field is used as the Endorsing Bank Item Sequence number in field 8 of the first record type 25. Assigned by the institution that creates the Check Detail Record	Convert From 9.37
	The numeric value of the Sequence field is used as the Endorsing Bank Item Sequence number in field 5 of the first record type 35. Assigned by the institution that creates the Return Check Record Valid Values: Space 0-9999999999999999 (16 bytes)	Write Return Write Return Item Write Return Find List

Field Name	Description, Valid Codes, and Comments	Function Use
Federal Format	Specify whether X9.37 file should conform to the Federal Reserve Board requirements. This will override the parameter settings for Encoding and ByteOrder. It will also convert all routing numbers to the FED requirement. This should NOT be used to generate files for IRD printing since the FED requirement and the ASC X9.90 requirements are different. Valid values: True – conforms to Federal Reserve Board **(Big Endian)** **Requirements - Encoding: EBCDIC** **Byte Order: Motorola** False (default) – Do not convert to Fed requirements	Convert From NSF - **chg to True** Write Return Write Return Item Write Return Find List
Encoding	Controls **the character type** of the output 9.37 file The input file is detected automatically and this field is to change the current value. Valid values: EBCDIC (default - Fed) ASCII	Convert From 9.37 Convert From NSF
Byte Order	Controls the **data** byte order of the output 9.37 file The data byte order is automatically detected/identified And this field is to change the current value. Valid values: Intel (default) (Little Endian) Motorola (Big Endian - is Fed) Note: Intel will display 50 00 00 00 Motorola will display 00 00 00 50 Reverses first 4 bytes in File Header 01 Record	Convert From 9.37 Convert From NSF Convert From CSV
TIFF Byte Order	Controls the TIFF order of the output 9.37 file The input file is detected automatically and this field is to change the current value. (Motorola is the only accepted value for Tough Tiff.) Valid values: Intel (default)	Convert From 9.37

Field Name	Description, Valid Codes, and Comments	Function Use
Source Rec. 61 Version	Property specifies the expected version of ***Record 61*** from source X9.37 files. The exact format of input Record 61 in X9.37 files varies according to bank requirements. See Appendix. Valid values: 1 - 2 char source of work 2 – 1 char source of work – No Record Indicator 3 – 1 char source of work – Record Indicator (Only Version 1 can be used for 180 format. Contact your Bank/Vendor for incoming formats)	Convert From 9.37 Convert To 180 Merge Cash Letter Merge Bundles
Dest. Rec 61 Version	Property specifies the expected version for the Destination ***Record 61*** format. The exact format of output Record 61 in X9.37 files varies according to bank requirements. See Appendix Valid values: See above Source Rec. 61 Version Other values can be used depending on credit format, adding a 40 or 20 to the original value) 41 – insert a credit record on a cash letter basis 21 – insert a credit record on a bundle basis (Only Version 1 can be used for 180 format. Contact your Bank/Vendor for format) Other special formats are available upon request. **NOTE: There have been many new formats added See Appendix for additional information**	Convert From 9.37 Merge Cash Letter Merge Bundles
Source Rec 68 Version	Property specifies the version of input Record 68 The exact format of Record 68 will depend on the Companion Document provided by the bank Valid values can be found in the Appendix 0 - Unknown - Format not specified 1 - Default937 - Variable length 4 - Default180 - 180 version - variable length 5 – Wells Fargo format - Fixed 80 length record 6 - 937 to Wells Fargo 10 - Same as default 14 - Format 14	Convert From 9.37 Convert To 180 Merge Cash Ltr Merge Bundle
Dest Rec 68 Version	**P**roperty specifies the expected output version of ***Record 68*** format. The exact format of Record 68 will depend on the Companion Document provided by the bank Valid values: See above See Appendix	Convert From 9.37 Merge Cash Ltr Merge Bundles

| Field Name | Description, Valid Codes, and Comments | Function Use |

Bundle Limit	Specifies the number of items in a bundle	Limit Bundles
Repair Threshold	How aggressively you want to repair image (value=6 is acceptable)	Convert From 9.37
Repair Max Image Size	Maximum size of images Fed will reject (value=140,000)	Convert From 9.37
Convert Images	**Valid values:** Checked = Input check images that are not compliant will be converted to bilevel Group 4 images that conform. 200/240 both should be acceptable Blank = The style will not be changed. This option is useful for those users that want to convert between ASCII and EBCDIC formats without disturbing the format of check images. **NOTE: This function will NOT convert JPEG to Tiff**	Convert From 9.37
Repair Images	Image background removal **(**Determine the Repair Threshold and Check Repair Images)	Convert From 9.37
Verify IRD Mode	Sets IRD mode which is more lenient than the default MICR mode.	Verify 937 Nsf/Blob
Convert Non-ABA to Credit	If set, causes check items with invalid ABA routing numbers to be converted to Record 61.	Convert From 9.37
Cores	Single core hyper-threaded CPUs should use a value of 2 for maximum performance. Valid values: 1 -16 depending upon the number of CPU cores available in the PC environment	Verify 937 Nsf/Blob
Enable Logging	Checking this field, produces an expanded Error Log dropping the .937 and appending .log to file name, ie. Extended log for Temp.nsf.937 would be temp.nsf.log	All
Enable Verbose Logging	Checking this field, produces an expanded Error Log dropping the .937 and appending .log to file name, ie. Extended log for Temp.nsf.937 would be temp.nsf.log	All

Field Name	Description, Valid Codes, and Comments	Function Use

Endorsement Options(Drop Down) **Used in conjunction with Endorse 9.37 Opinon and gives parameters for adding** an endorsement on the back of each check in the X937 file. **Endorse 937**

Bank Name: the second line of the endorsement
If this line is left blank, the organization name field data from the 9.37 file will be used.

Bank Routing: The first line of the endorsement. If this line is left blank, the organization routing field data from the 9.37 file will be used.

Deposit Account Number: The third line of the endorsement. If this line is left blank, the BOFD account number from the 9.37 file will be used.

Deposit Name: The fourth line of the endorsement.

Font Style: Selects the font type used for the endorsement. If a null string is specified, the font defaults to Times New Roman. Fonts should be specified by exact names, such as Helvetica or Courier New

Drawing Options: Selects drawing and format options used in the endorsement. Must use a string in the field:

j= sets the *justification* to left(l), centered(c), or right (r) default is centered, example
j=r would right justify

s= sets the *font point size*. Valid values are from 4 to 48. If unspecified, the default font size is 12, example:
s=10 would create a size 10 font

w= sets the *weigtht or darkness of the text*. '400' indicates normal darkness, and '800' indicates bold darkness. Default is '400'.

NOTE: multiple options should be added together without spaces, i.e.
j=cs=10w=200

Endorse Height (in): Height of endorsement area. A value of 0 specifies that a default value will be used.

Endorse Width (in): Width of endorsement area. A value of 0 specifies that a default value will be used.

Field Name	Description, Valid Codes, and Comments	Function Use

Bottom Offset (in): The distance from the bottom of the check to the endorsement area. A value of 0 specifies that a default value should be used. 0.01 should be used to specify no offset

Right Offset (in): The distance from the right side of the check to the endorsement area. A value of 0 specifies that a default value should be used. 0.01 should be used to specify no offset

Deposit/Payee Endorse: If checked, the **default** endorsement layout will be vertical, in the right 1.5 inches of the check above 0.625 inches from the bottom.

BOFD Endorse: If checked, the default endorsement will be horizontal, between 3 inches from the left side of the check and 1.8 inches from the right side of the check, above 0.625 inches from the bottom.

Bank/Transit Endorse: If checked, the default endorsement layout will be horizontal in the left 3 inches of the check above 0.625 inches from the bottom

Horizontal Box Endorse: If checked, the default endorsement will be horizontal, and the default location values will locate the text in the upper left corner.

Vertical Box Endorse: If checked, the default endorsement will be vertical, and the default location values will locate the text in the middle of the right side.

Destination Credit (Drop Down) **Used in conjunction with Dest Rec. 61 version and a**dds Destination credit information for credit records (R61 or R25) that are automatically inserted. ConvertFrom937

Depositor Routing: To be included on the credit image

Account Number: A numeric value assigned for use with ***Credit*** records. 20 bytes - system will right justify space fill.

Depositor Name: To be included on the credit image

Depositor Date: To be included on credit image

Depositor Routing x(in) - y(in): X and Y positions of Depositor Routing number image in inches * 100

Account x(in) - y(in) - X and Y positions of account # on an image in inches * 100

Depositor Name x(in) y(in:) X and Y positions of Deposit Name image in inches * 100

Depositor Date x(in) y(in): X and Y positions of Deposit Date image in inches * 100

Amount x(in) y(in): X and Y positions of Amount image in inches * 100

Item Count x(in) y(in): X and Y positions of Item Count image in inches * 100

Field Name	Description, Valid Codes, and Comments	Function Use

Sort (Drop Down) ` `	**Parameters that work in conjuction with Sort functions**	Sort SortBofD
File Set Index	Used in conjunction with Sort to generate an extension on the file name for sorted records to group items by Payor Routing (Record 25, Field 4) of intermediate files in the WD. Also creates AMP.TOF containing a list of these intermediate files. Valid Values:0 to N, (N < 937). Unique index code (0 to N) for the file being sorted. First call to any of the Sort methods must pass in 0.	
nUp	The number of cuts between cuts. Used in preparation for IRD printing. Valid Values: integer	Cutter Sort
Sort Dups Start Item	Start number of item, if any	
Sort MICR fields sort order	Extracts items based on MICR fields	
Route:	Low-High, Off, High-Low	
Account:	Low-High, Off, High-Low	
Check:	Low-High, Off, High-Low	
Amount:	Low-High, Off, High-Low	

AX9LIB Functionality Test

Entry into AX9 Demo Screen:

Navigate to c:\Program Files\AllMyPapers\ax9lib\bin
Double Click AX9DemoNet
AX9 Demo Screen will appear

For testing functionality, set up the temporary files.

Click on Working Directory and set up ***2*** temporary directories:
c:\tempdemotest
c:\tempax9

(For functionality testing, we will place the working files in *tempdemotest* and the source in *tempax9*.)

Note: Executing the first example below (Convert From Nsf) will convert the sample Nsf file sent with the system and create a X9.37 file that can be used to test the remainder of the functions.

Read/Write Functions

Note: Review Appendix A for the various file formats AX9LIB reads and writes.

Convert From Nsf (Read Nsf/Sit Write 937)

The Convert From NSF function will read the Normalized Scanner Format (NSF) file with single image Tiff files sent with the system and construct a X9.37 file in the Working Directory. The function will auto-detect the various source NSF formats and process accordingly.

As stated, the function will automatically detect NSF or NSF2 file formats;

NSF format has a comma separated item for each X9.37 MICR line field. NSF can be generated by reading a X9.37 file and using Convert to NSF/SIT or can be created using Notepad.

NSF2 format has a single field for the whole MICR line and a translation table. NSF2 is typiclly what would come from a scanner. There is no AX9Lib function to generate a NSF2 format.

When the file is NSF/SIT, and you are using the ConvertFromNSF function, all of the Single Image TIFF or JPEG files referenced in the source NSF file must exist and be valid TIFF or JPEG files for this function to successfully finish. The extension for a JPEG file must be ".JPG" or ".jpg". A TIFF file may have any extension.

When the input file is NSF BLOB, the BLOB file must exist and contain all the referenced data for this function to successfully finish.

The ECE Institution Item Sequence Number may be included on the NSF line after the other data. If the NSF data does not have Item Sequence Number data, then the numeric value in the Sequence number field will be incremented and used as the ECE Institution Item Sequence Number for entry in field 8 of the first record type 25. (The test file has Sequence Number Data, so it is not necessary to enter this field.)

The Federal Format parameter may be set to ***True*** to indicate that the X9.37 file should conform to the Federal Reserve Board requirements. This will override the parameter settings for X9Conversions. If is it set, AX9Lib will also convert all routing numbers to the FED requirement. **The Federal Format field should NOT be set to True to generate files for IRD printing since the FED requirement and the ASC X9.90 requirements are different.**

Example 1: Read a NSF file and convert to output X9.37 file entering a valid Destination Routing Number and Creator Routing Number as input parameters:

Screen Parameters:

File Name: NSf Input file

c:\program files\allmypapers\ax9lib\examples\temp.nsf

(Nsf file included with install libraries.)

Working Directory: Working file created prior to starting for output file

c:\tempax9

Destination Routing: 123456780 - Will set the following records/fields,

Record 1, field 4
Record 10, field 3
Record 20, field 3

There is no fields in the input NSF for these fields and must be entered via these parameters to be valid.
Blank will cause these fields to be invalid (see X9Viewer screen)

Creator Routing: 987654320 Will set the **ECE** Institution Routing number in the Following records/fields: Record 1, field 5
Record 10, field 4
Record 20, field 4

There is no fields in the input NSF for these fields and must be entered via these parameters to be valid.
Blank will cause these fields to be invalid (see X9Viewer screen)

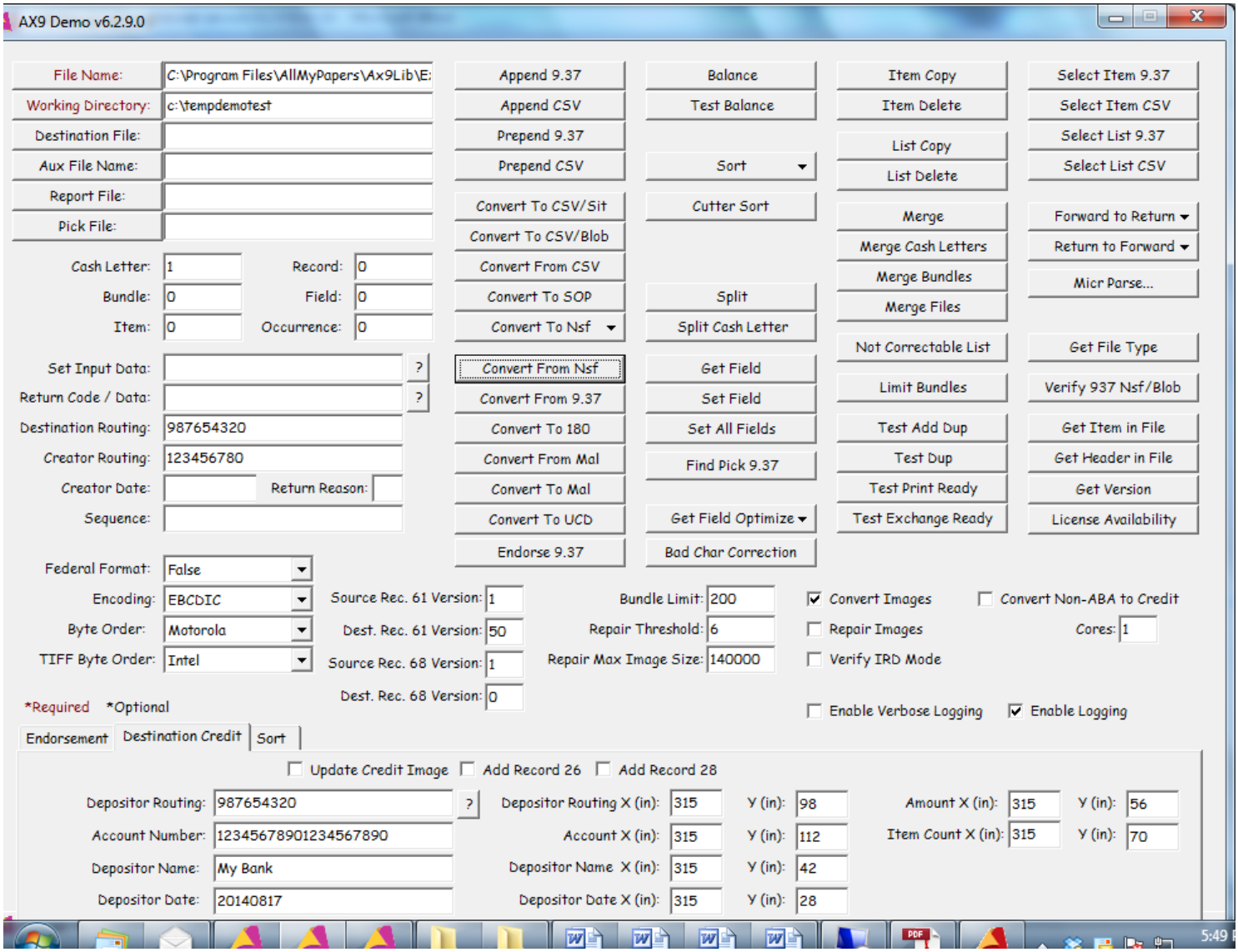
Sequence: 0 (if don't have a value in the NSF file then AX9Lib increments to 1 for 1st Record 25, Field 8)

Federal Format: optional default

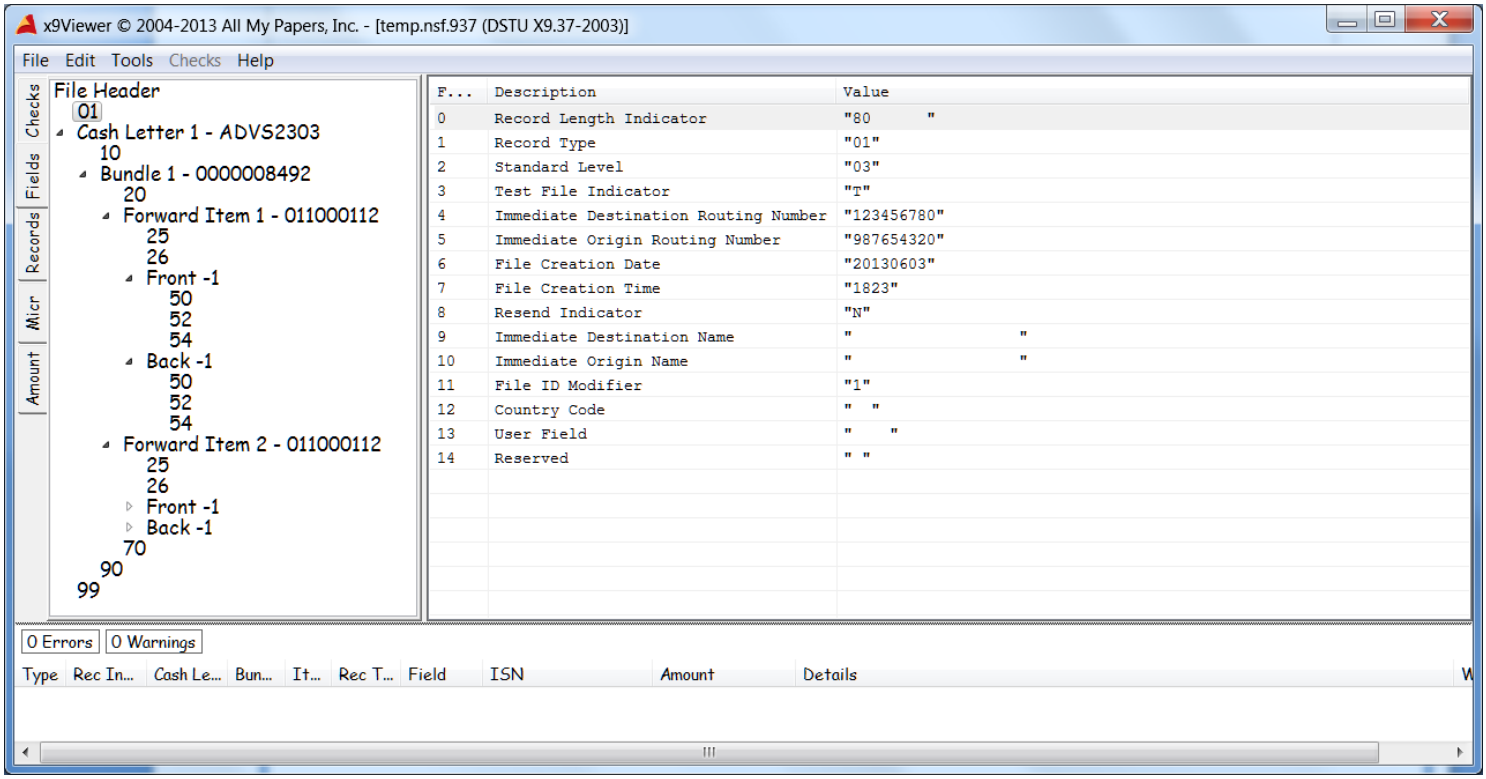
The output files in the Working Directory will have the same name as the original source file appended with the extension .937.

c:\tempax9\temp.nsf.937 (View in X9Viewer)
c:\tempax9\temp.nsf.937 TOC.log (View in Notepad)
c:\tempax9\temp.nsf.937 TOC (View in Notepad)
c:\tempax9\temp.nsf.937 TOC.SUM (View in Notepad)

Each set of output files will have similar run-time logs and Summary files.


AX9 Demo v6.2.9.0
File Name:
C:\Program Files\AllMyPapers\Ax9Lib\E:
Working Directory:
c:\tempdemotest
Destination File:
Aux File Name:
Report File:
Pick File:
Cash Letter:
1
Record:
0
Bundle:
0
Field:
0
Item:
0
Occurrence:
0
Set Input Data:
Return Code / Data:
Destination Routing:
987654320
Creator Routing:
123456780
Creator Date:
Return Reason:
Sequence:
Federal Format:
False
Encoding:
EBCDIC
Byte Order:
Motorola
TIFF Byte Order:
Intel
*Required
*Optional
Source Rec. 61 Version:
1
Dest. Rec. 61 Version:
50
Source Rec. 68 Version:
1
Dest. Rec. 68 Version:
0
Bundle Limit:
200
Repair Threshold:
6
Repair Max Image Size:
140000
Convert Images
Repair Images
Verify IRD Mode
Convert Non-ABA to Credit
Cores:
1
Enable Verbose Logging
Enable Logging
Append 9.37
Append CSV
Prepend 9.37
Prepend CSV
Convert To CSV/Sit
Convert To CSV/Blob
Convert From CSV
Convert To SOP
Convert To Nsf
Convert From Nsf
Convert From 9.37
Convert To 180
Convert From Mal
Convert To Mal
Convert To UCD
Endorse 9.37
Balance
Test Balance
Sort
Cutter Sort
Split
Split Cash Letter
Get Field
Set Field
Set All Fields
Find Pick 9.37
Get Field Optimize
Bad Char Correction
Item Copy
Item Delete
List Copy
List Delete
Merge
Merge Cash Letters
Merge Bundles
Merge Files
Not Correctable List
Limit Bundles
Test Add Dup
Test Dup
Test Print Ready
Test Exchange Ready
Select Item 9.37
Select Item CSV
Select List 9.37
Select List CSV
Forward to Return
Return to Forward
Micr Parse...
Get File Type
Verify 937 Nsf/Blob
Get Item in File
Get Header in File
Get Version
License Availability
Endorsement
Destination Credit
Sort
Update Credit Image
Add Record 26
Add Record 28
Depositor Routing:
987654320
Account Number:
12345678901234567890
Depositor Name:
My Bank
Depositor Date:
20140817
Depositor Routing X (in):
315
Y (in):
98
Account X (in):
315
Y (in):
112
Depositor Name X (in):
315
Y (in):
42
Depositor Date X (in):
315
Y (in):
28
Amount X (in):
315
Y (in):
56
Item Count X (in):
315
Y (in):
70

When you view the file in the X9Viewer (see below), you will notice it is error free. If the routing numbers had not been included in the input parameters, the X9Viewer would have indicated the routing numbers did not conform and it would be necessary to return to AX9Lib and enter the destination and creating routing numbers.



Note: If you do not delete your work files, they will just be written over by the new file, which is what will happen in the next example.

Example 2: Read a NSF file and generate an output X9.37 file with a Record 28 included. The first line of the NSF file is options and a field 6 needs to be added which accepts a value of "28" indicating create a Record 28 in the X937 file. The contents of Record 28 is derived from the input parameters and the Record 25 and 26 data.

Example NSF used to create a Record 28 in the output file:

NSF,"C:\All My Papers\X9.37\Copy of Fed\ForwardSample[1].txt",0,0,"Copyright 2005 All My Papers","**28"**

Screen Parameters:

Working Directory: Working file created prior to starting for output file
c:\tempax9

File Name: Nsf Input file
c:\program files\allmypapers\ax9lib\examples\temp.nsf
(include the "28" value described above)

Destination Routing:
123456780

Creator Routing:
987654320

Federal Format: (changes from Little Endian to Big Endian)
True

The output file in the Working Directory will have the same name as the original appended with the extension .937.

c:\tempax9\temp.nsf.937 **(View in X9Viewer)**

This file is now a valid 937 file. There are no error messages when viewed in X9Viewer.

Convert From 9.37 (Read X9.37 and Write X9.37)

This function will read the input X9.37 file specified by File Name and output an X9.37 file in the Working Directory with the name fileName.937.

This function has multiple uses depending on the parameters used as input.

1. **The primary purpose of this function is to correct existing X9.37 files and/or convert to a different format of X9.37 as required by an exchange partner or for IRD Printing.**

   This function will convert non Annex F compliant X9.37 files to files that are compliant. Annex F rules deal mainly with the style of TIFF (Tagged Image File Format) check images embedded within the X9.37 file. For example, Annex F compliant TIFF files are Group 4 encoded at a resolution of 200 DPI, contain a single strip of image data, and have little Endian bit order within the data bytes. (See AX9Lib API Reference Guide.)

   The Conversion fields (Encoding, Byte Order, TIFF Byte Order) for the input file are all detected automatically by AX9Lib, but can be modified or controlled for the output file by using the screen fields. For example, this function is used to control formats acceptable to the FRB. Other conversions will happen automatically. The system automatically converts the following when the specified screen fields are set:

   a. Images are converted to Annex F compatible ***if*** the Convert Images box is checked. If set to nonzero, input check images that are not Annex F compliant will be converted to bilevel Group 4 images that conform to the Annex F. If set to zero, the compressed image style will not be changed. This option is useful for those users that want to convert between ASCII and EBCDIC formats without disturbing the format of check images.

   b. Record 61 (if present) is converted to the format designated by the Destination version of record 61. In many cases this is required to be an IRD Print compatible Record 61 (Version 1).

      Note: If the 937 file has a Record 61, we need to identify what version is being using for the input file and the output file using the parameters Source Record 61 version and Dest Record 61 Version. It is up to the user to identify the input Record 61 format. (Appendix C - Record 61, but there are many new specialty versions specified by the various companion documents provided by receiving banks. Contact AMP if the format you need is not identified.)

   c. Check items with non-valid ABA routing numbers are converted to Record 61 if Convert Non-ABA to Credit is set.

   d. X9.37 is converted to UCD 187 compliant if Dest. Rec. 61 =15. (Note: As of 2010, a new function called ConvertToUCD can be used for this function.)

2. ***A secondary purpose of the Convert From 9.37 function is to auto insert Credit Records into bundles or cash letters when none existed before. The inserted records will conform to the different Record 61 Versions.***

   To cause auto insertion of Record 61 on a *Bundle* basis, increase the Version number in the Dest. Rec. 61 version screen field by 20 over the standard conversion number.

   To insert on a *Cash Letter* basis, the Destination conversion type value is increased by 40.

   For example:
   Dest. Rec 61 Version = 21 will insert AMP Version 1 Record 61 on a Bundle basis.
   Dest. Rec 61 Version = 41 will insert AMP Version 1 Record 61 on a Cash Letter basis.

   Source Rec. 61 Version = 1-3
   Note: There are many variations of the Record 61. AMP has created specialty formats in regard to certain variations. If they are not included in Appendix C, contact All My Papers Support for specifics.

3. ***Another purpose of the Convert From 9.37 function is to delete a variety of records from the source file. Note: This is records, not items.*** By setting fields in the Set Input Data field, you can select all records you want to delete Note: Deleting the first record of an item could be a catastrophe if using a production file, e.g., 25, 31, 61, but most records can be deleted to create an invalid file for testing. Remember there are **multiple records** for each item.

4. ***The last use creates a pick list of items that have bad images***. This can then be used in the Not Correctable function. (Under Construction - Will create a new picklist file that only has the 937 items that are not corrrectable in a ConvertFrom937 operation.)

Example 1 - Read a X9.37 file, convert the input X9.37 to the EBCDIC/Motorola, convert the images, convert the record 61s (if any) from Version 2 to version 1, change the Account # number and create an output X9.37 file. It will display the number of images out of compliance in the Return Code/Data field on the screen:

Screen Parameters:

Working Directory: Working file created prior to starting for output file
c:\tempdemotest

File Name: X9.37 Input file
c:\tempax9\temp.nsf.937

Sequence: Institution Item Sequence (Record 61AMP 1 version, Field 8)
22222222

Account Number: specifies account number assigned for use with credit record (Record 61,AMP 1 version, Field 6 - Posting Bank Account

Number -in Destinations Credit Drop Down)
12345678901234567890

Encoding: Specifies whether ASCII or EBCDIC encoding for the output file.
EBCDIC

Byte Order: The byte order of the output file, Intel or Motorola (Big Indian)
MOTOROLA

TIFF Byte Order: The byte order for the output . TIF file
Intel

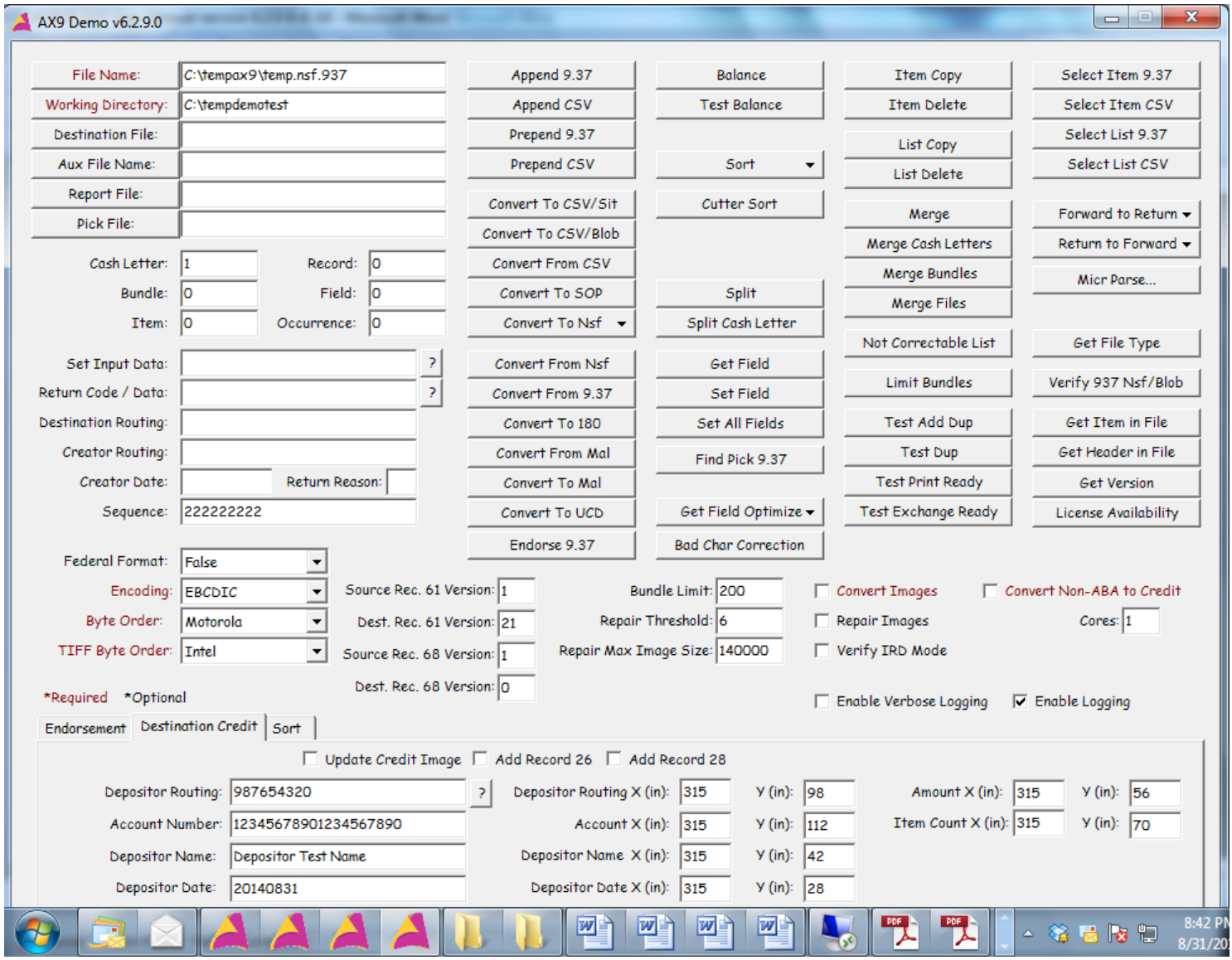
Source Rec. 61 Version: Input version is an AMP1
1

Dest. Rec. 61 Version: Output version should be a 21
21

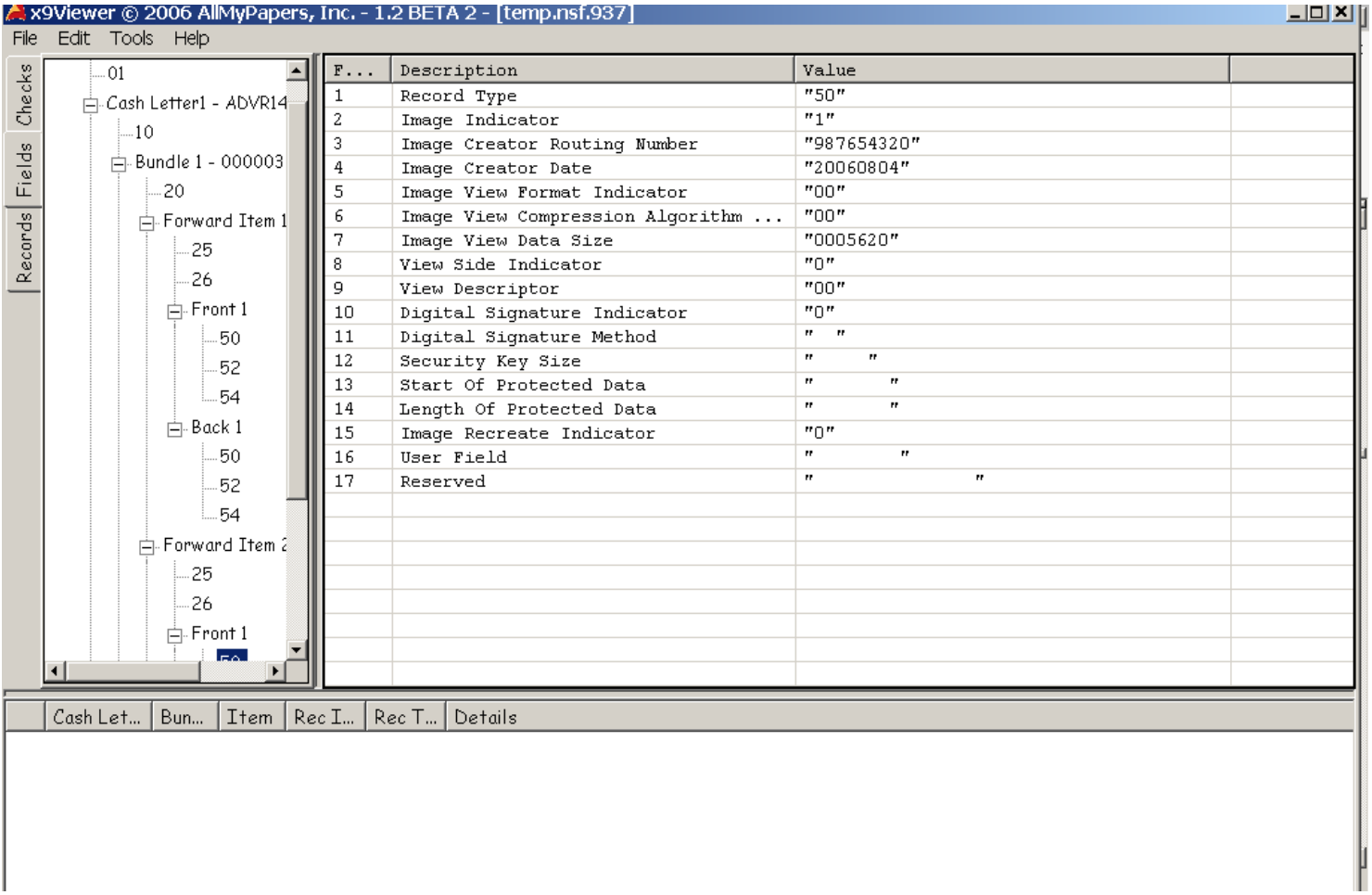
Convert Images: Checked - input check images that are not compliant will be converted to bilevel Group 4 images that conform to the Annex F. If left blank, the compressed image style will not be changed. This option is useful for those users that want to convert between ASCII and EBCDIC formats without disturbing the format of check images. Need an Amplib license.
Note: Will not convert JPEG to TIFF.

Convert Non-ABA to Credit: if checked, causes items with invalid ABA routing Number to be converted to Record 61's (credits)
Output file in the Working Directory.

c:\tempdemotest\temp.nsf.937 **(View file in X9Viewer)**


AX9 Demo v6.2.9.0
File Name:
C:\tempax9\temp.nsf.937
Working Directory:
C:\tempdemotest
Destination File:
Aux File Name:
Report File:
Pick File:
Cash Letter:
1
Record:
0
Bundle:
0
Field:
0
Item:
0
Occurrence:
0
Set Input Data:
Return Code / Data:
Destination Routing:
Creator Routing:
Creator Date:
Return Reason:
Sequence:
222222222
Federal Format:
False
Encoding:
EBCDIC
Byte Order:
Motorola
TIFF Byte Order:
Intel
*Required
*Optional
Append 9.37
Append CSV
Prepend 9.37
Prepend CSV
Convert To CSV/Sit
Convert To CSV/Blob
Convert From CSV
Convert To SOP
Convert To Nsf
Convert From Nsf
Convert From 9.37
Convert To 180
Convert From Mal
Convert To Mal
Convert To UCD
Endorse 9.37
Balance
Test Balance
Sort
Cutter Sort
Split
Split Cash Letter
Get Field
Set Field
Set All Fields
Find Pick 9.37
Get Field Optimize
Bad Char Correction
Item Copy
Item Delete
List Copy
List Delete
Merge
Merge Cash Letters
Merge Bundles
Merge Files
Not Correctable List
Limit Bundles
Test Add Dup
Test Dup
Test Print Ready
Test Exchange Ready
Select Item 9.37
Select Item CSV
Select List 9.37
Select List CSV
Forward to Return
Return to Forward
Micr Parse...
Get File Type
Verify 937 Nsf/Blob
Get Item in File
Get Header in File
Get Version
License Availability
Source Rec. 61 Version:
1
Dest. Rec. 61 Version:
21
Source Rec. 68 Version:
1
Dest. Rec. 68 Version:
0
Bundle Limit:
200
Repair Threshold:
6
Repair Max Image Size:
140000
Convert Images
Repair Images
Verify IRD Mode
Convert Non-ABA to Credit
Cores:
1
Enable Verbose Logging
Enable Logging
Endorsement
Destination Credit
Sort
Update Credit Image
Add Record 26
Add Record 28
Depositor Routing:
987654320
Account Number:
12345678901234567890
Depositor Name:
Depositor Test Name
Depositor Date:
20140831
Depositor Routing X (in):
315
Y (in):
98
Account X (in):
315
Y (in):
112
Depositor Name X (in):
315
Y (in):
42
Depositor Date X (in):
315
Y (in):
28
Amount X (in):
315
Y (in):
56
Item Count X (in):
315
Y (in):
70

We have changed the Byte Order to Motorola, so the field ***Image View Data Size*** has changed from 0007036 to 0005620.



Example 2 - Read a X9.37 file, and add a credit Record 61 in the Wells Fargo format. (See document AX9Lib Creating WFED Format Files in AX9LIB Help file for complete instructions on creating WFED files.)

Screen Parameters:

Working Directory: Working file created prior to starting for output file
c:\test

File Name: X9.37 Input file
c:\tempax9\R68 V4 multiple bundles changed.csv.937

Destination File: WFD (936) Output file
c:\tempdemotest\joan test 936

Aux File Name: $$Add header addendum (Input - add header to file)
c:\EMON_AuxFileName.txt

Encoding: Specifies whether ASCII or EBCDIC encoding for the output file.
EBCDIC

Byte Order: The byte order of the output file, Intel or Motorola (Big Indian)
Intel

TIFF Byte Order: The byte order for the output . TIF file
Intel

Source Rec. 61 Version
1

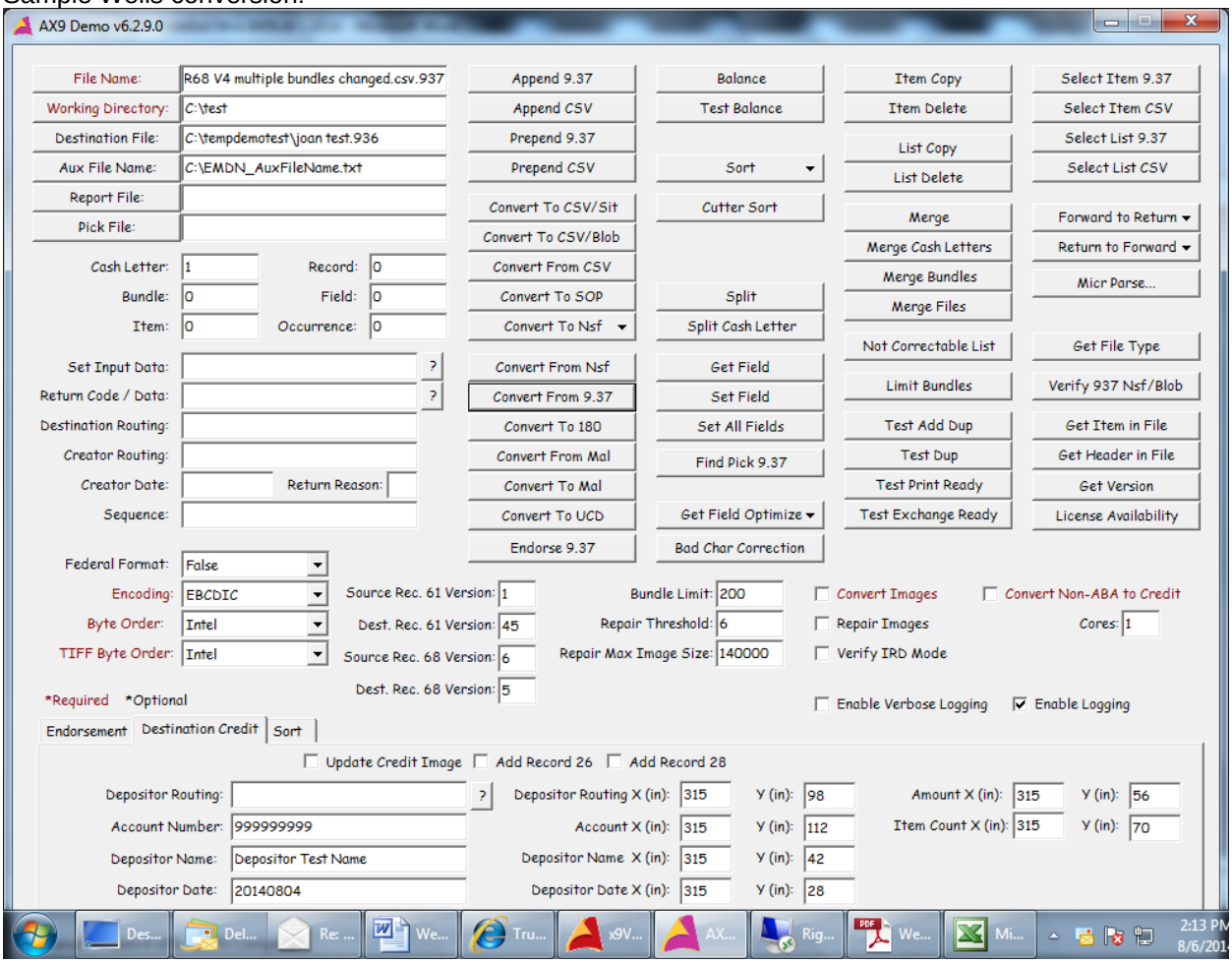
Dest. Rec. 61 Version
45

(If Record 68s are used, these parameters will also need to be set)
Source Rec. 68 Version
6

Dest. Rec. 68 Version
5

This will create a WFED .936 file with a Record 61, carrying the R68 through, and adding the Wells Header record.

Sample Wells conversion:



Example 3 - Read a X9.37 file, and delete all the Record 52's in the file. This will create a non-image file that can be used for testing. Note: This is the only function available that deletes Records. Other functions delete Items:

Screen Parameters:

Working Directory: Working file created prior to starting for output file

c:\tempdemotest

File Name: X9.37 Input file

c:\tempax9\temp.nsf.937

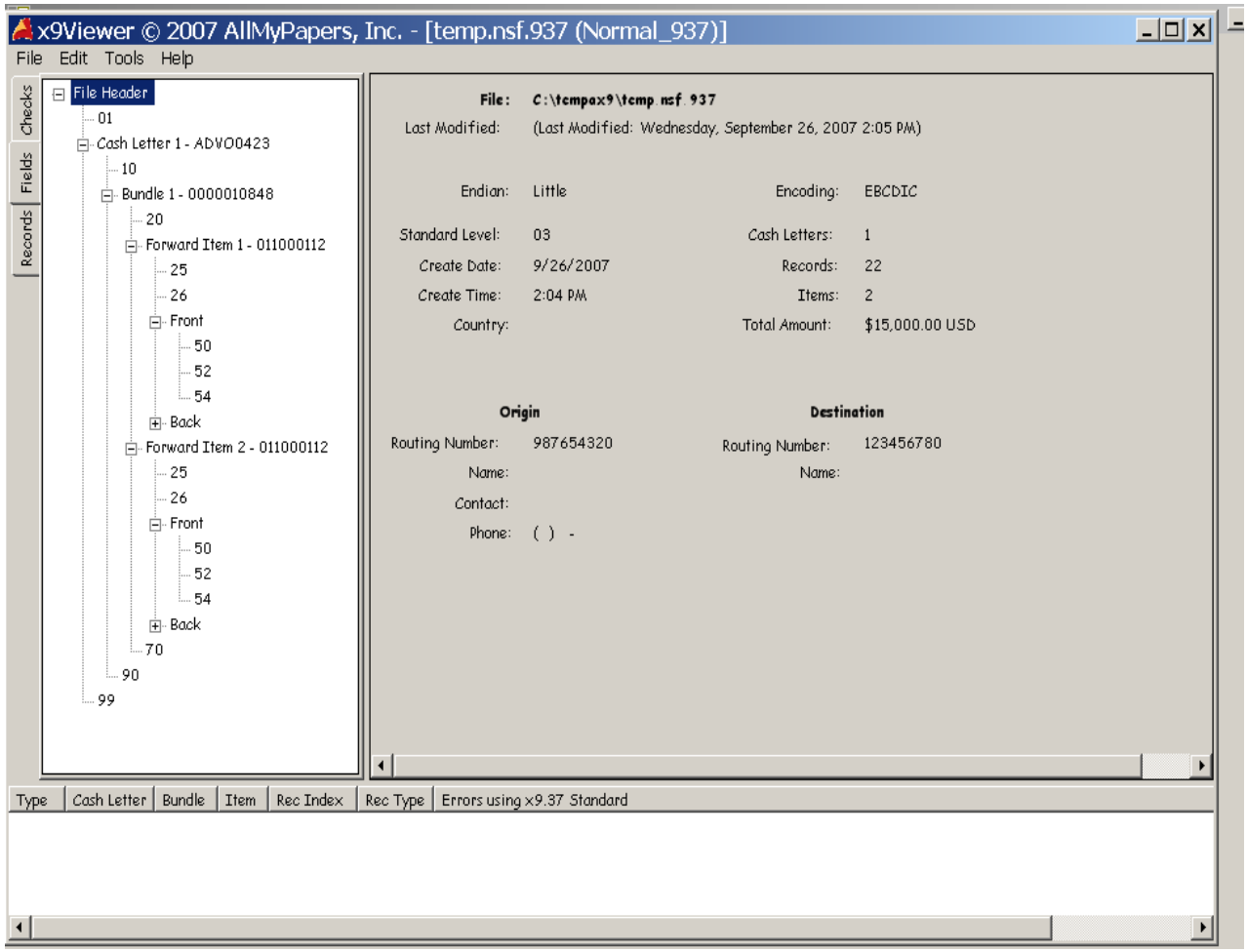
Set Input Data: enter records to delete:

52

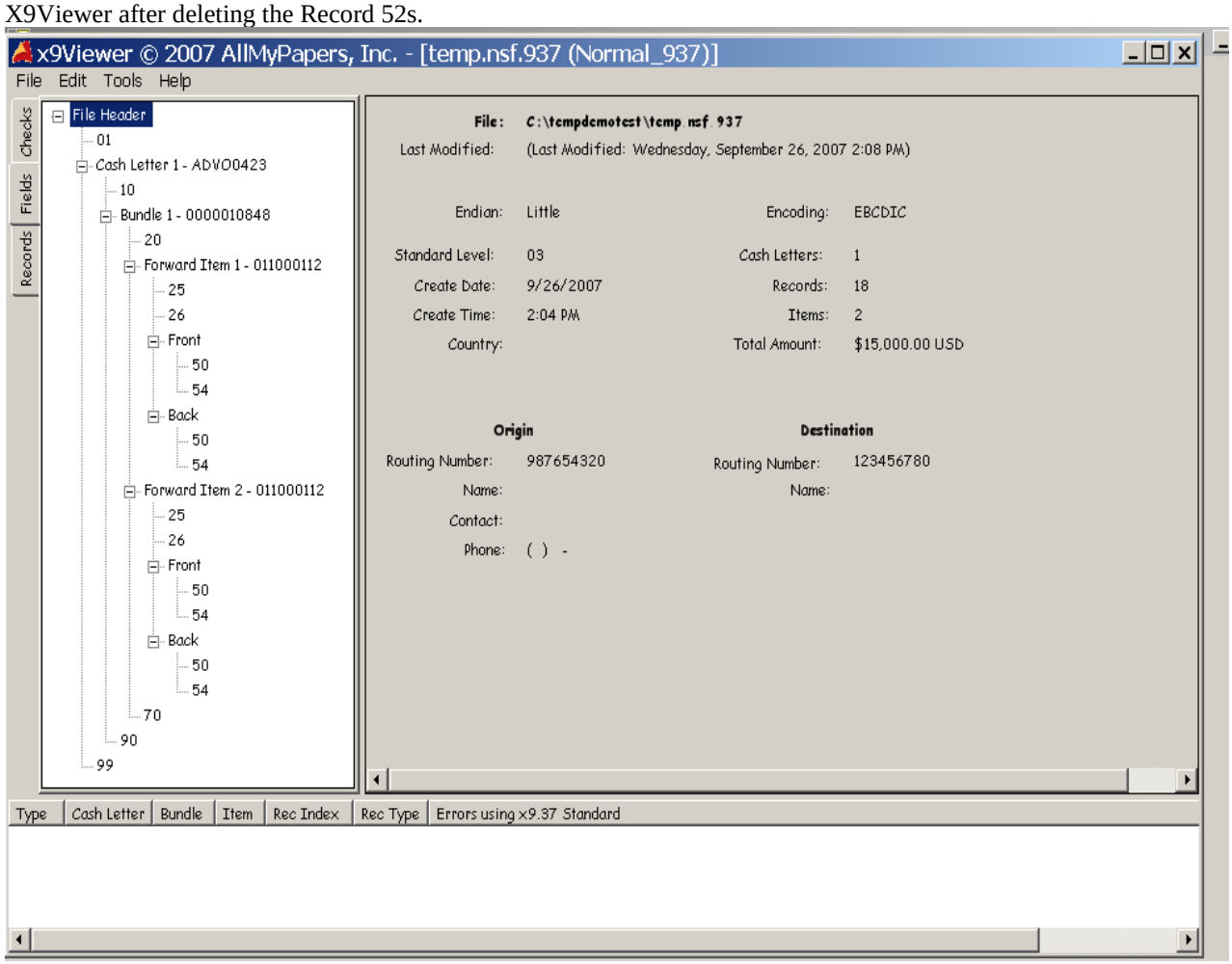
Output file in the Working Directory.

c:\tempdemotest\temp.nsf.937 **(X9Viewer will indicate missing record and re-insert when viewed)**

X9Viewer before deletion of all record 52s.



X9Viewer after deleting the Record 52s.



Example 4 - Read a X9.37 file, and create a pick list of items that have bad images. The output file can be used in conjunction with Not Correctable List function.

Screen Parameters:

Working Directory: Working file created prior to starting for output file
c:\tempdemotest

File Name: X9.37 Input file
c:\test files\non correctable images.937

Pick File(output): This is where the list of items with bad images will be placed
c:\tempdemotest\image pick list.lst

Convert From CSV (Read CSV Write X9.37)

This function will read the input CSV file File Name field and output an X9.37 file in the Working Directory.

This function will accept CSV files containing all the field information desired in a resulting X9.37 file. Each field must have the length consistent with the same field in the X9.37 record. The variable length information for a field is supplied by the named file. **Note: Creator and Destination Routing numbers will be taken from the input file.**

The Convert From CSV function can create a partial X9.37 file when presented with less than a full set of records. For example, only records in the range 25-54 will create all the records for a specific item. This file can then be inserted into a X9.37 file with the Append/Prepend functions.
Example to read a CSV file and convert to output X9.37 file:
Screen Parameters:

Working Directory: Working file created prior to starting (output file will be placed)
c:\tempax9

File Name: CSV Input file
c:\program files\allmypapers\ax9lib\examples\temp.csv

The output file in the Working Directory will have the same name as the original appended with the extension .937.
c:\tempax9\temp.csv.937 **(View file in X9Viewer)**

Convert To SOP (Read X9.37 Write SOP)

This function will read the input X9.37 File Name field and output an SOP IV.8 formatted file in the Working Directory.

Note: The SOP IV.8 file format is described with limited detail in a FRB document. This function requires **the X9.180 license feature bit.**

Example to read a X937 file and convert to output SOP file:
Screen Parameters:

Working Directory: Working file created prior to starting (output file will be placed)
c:\tempax9

File Name: X9.37 file Input file
c:\tempdemotest\temp.nsf.937

The output file in the Working Directory will have the same name as the original appended with the extension .937.
c:\tempax9\temp.csv.937.sop **(View file in Notepad)**

Convert to CSV/SIT

Read X9.37 WriteCsv and multiple single image TIFF files.
The Convert to CSV/SIT function will read an X9.37 file entered in File Name and generate an output CSV file and also generate single-imagte TIFF files in the Working Directory.

The function will create a partial CSV file when presented with less than a full set of records in the input X9.37 file. For example, only records in the range 25-54 will create all the records for a specific item. This partial file can then be inserted into a X9.37 file with the Append/Prepend functions.

Note: If the 937 contains a Record 61, you need to identify what version/format of the Record 61 you are using by using the fields Source Record 61 version and Dest Record 61 version. Both fields would be populated with the same value.

Example to read a X9.37 file and convert to output CSV/SIT file:

Screen Parameters:

Working Directory: Working file created prior to starting for output file
c:\tempdemotest

File Name: X9.37 Input file
c:\tempax9\temp.nsf.937

The output file in the Working Directory will have the same name as the original appended with the extension .csv. and TIFF images.

c:\tempdemotest\temp.nsf.937.csv	**(Excel spreadsheet file)**
c:\tempdemotest\0000007.tif	**(Images – double click)**
c:\tempdemotest\00000010.tif	**(Images – double click)**

Convert to Nsf/Sit (Read 937 Write Nsf/Sit)

The Convert to Nsf/Sit function will read a X9.37 file entered in File Name and **generate a Normalized Scanner Format (NSF) output file** in the Working Directory along with all the **Single Image TIFFs (SIT)** contained in File Name.

The TIFF images will be given their original names as referenced in the X9.37 file. The NSF file will have the same name as the original File Name appended with a .NSF extension. For more information on the Normalized Scanner Format, see Appendix C.

Example to read a X9.37 file and convert to output Nsf/Sit file:

Screen Parameters:

Working Directory: Working file created prior to starting for output file
c:\tempdemotest

File Name: X9.37 Input file
c:\tempax9\temp.nsf.937

The output file will have the same name as the original appended with the extension .nsf.

c:\tempdemotest\temp.nsf.937.nsf (Double Click to view – Notepad)

The output file in the Working Directory will have the same name as the original appended with the extension .nsf and will contain the TIFF images.

c:\tempdemotest\temp.nsf.937.nsf (Excel spreadsheet file)
c:\tempdemotest\0000007.tif (Images – double click)
c:\tempdemotest\00000010.tif (Images – double click)

Convert To CSV/Blob (Read X9.37 Write CSV)

The Convert To CSV/Blob function will read a X9.37 file using a pointer to the image data, entered in File Name and generate an output CSV file in the Working Directory with the name File Name name.CSV. The name of the multi image blob file is stored in the CSV header record and it is NOT repeated with each line that uses it. A "Blob" is an unformatted concatenation of data.

The function will create a partial CSV file when presented with less than a full set of records in the input X9.37 file.

For example, only records in the range 25-54 will create *all* the records for a specific item. This partial file can then be inserted into a CSV file with the Append/Prepend function.

Example to read a X9.37 file and convert to output CSV file:

Screen Parameters:

Working Directory: Working file created prior to starting for output file
c:\tempdemotest

File Name: X9.37 Input file
c:\tempax9\temp.nsf.937

The output file in the Working Directory will have the same name as the original appended with the extension .csv.

c:\tempdemotest\temp.nsf.937.csv (Excel spreadsheet file)

Convert To NSF/Blob (Read 937 Write NSF)

The Convert To NSF/Blob function will read a X9.37 file entered in File Name and generate an output NSF (Normalized Scanner Format) file in the Working Directory. The NSF file will have the same name as the original File Name name appended with a .NSF extension.

This function is very fast because it builds a pointer to the image data in the NSF Blob format and does NOT copy the images from the original source X9.37 to individual files.

Example to read a X9.37 file and convert to output NSF file:

Screen Parameters:

Working Directory: Working file created prior to starting for output file
c:\tempdemotest

File Name: X9.37 Input file
c:\tempax9\temp.nsf.937

The output file in your Working Directory will contain the .nsf extension.

c:\tempdemotest\temp.nsf.937.nsf (Double Click to view – Notepad)

Convert to 180 (Read X9.37 Write 180)

A special license is required for this feature.

The Convert to 180 function will read an X9.37 file named File Name and generate an output X9.100-180 formatted file in the Working Directory with the original named File Name appended with .180.

The function will support a record 61 AMP version 1 if specified. The record61Version parameter will identify the format of the source record 61.

Warning: The X9.100-180 file conforms to the Second Ballot format of X9.100-180. This file is substantially different than previous non balloted reversions of the standard.

Warning: This function requires the X9180 license feature bit.

Example to read a X9.37 file and convert to output 180 file:

Screen Parameters:

Working Directory: Working file created prior to starting for output file
c:\tempdemotest

File Name: X9.37 Input file
c:\tempax9\temp.nsf.937

Source Rec. 61 Version
2

Dest. Rec. 61 Version
1

The output file in the Working Directory will have the same name as the original appended with the extension .180.

c:\tempdemotest\temp.nsf.937.180 **(X9Viewer if licensed)**

Convert to MAL (Read X9.37 Write MAL)

The Convert to MAL function will read an X9.37 file named File Name and generate an output MAL formatted file in the Working Directory with the original named File Name appended with .MAL

A MAL file is the Fed Payer Services ASCII file of 937 data with a non-standard header format.

Example to read a X9.37 file and convert to output MAL file:

Screen Parameters:

Working Directory: Working file created prior to starting for output file
c:\tempdemotest

File Name: X9.37 Input file
c:\tempax9\temp.nsf.937

The output file in the Working Directory will have the same name as the original appended with the extension .MAL.
c:\tempdemotest\temp.nsf.937.MAL (Notepad)

Convert From MAL (Read MAL Write X9.37)

The Convert From MAL function will read a MAL file named File Name and generate an output X9.37 formatted file in the Working Directory with the original named File Name appended with .937

A MAL file is the Fed Payer Services ASCII file of 937 data with a non-standard header format.

Example to read a MAL file and convert to output X937 file:

Screen Parameters:

Working Directory: Working file created prior to starting for output file
c:\tempdemotest

File Name: X9.37 Input file
c:\tempdemotest\temp.nsf.937.MAL

The output file in the Working Directory will have the same name as the original appended with the extension .X937.

c:\tempdemotest\temp.nsf.MAL.937 (X9Viewer)

Convert To UCD (Read X937 Write UCD (187)

Note: a special license is needed for this function.

The Convert To UCD function will read a 937 file named File Name and generate an output UCD formatted file in the Working Directory with the original named File Name appended with .937. NOTE: 7/2014 Normally UCD doesn't have R61's but most new companion documents are including - latest is AMP2 version. If you need to set fields in the R61, use =2 in both DEST and Source version 61 fields that indicates your input file is an AMP2 and it need to remain an AMP2. Then Balance the file.

Example to read a X937 file and convert to output UCD file:

Screen Parameters:

Working Directory: Working file created prior to starting for output file
c:\tempax9

File Name: X9.37 Input file
c:\tempdemotest\temp.nsf.937

The output file in the Working Directory will have the same name as the original.

c:\tempax9\temp.nsf.937 (X9Viewer)

Editing Functions

This set of functions is concerned with creating new files based on edits from the original X9.37 files. The result of the function is a fully formed X9.37 file.

It will contain all the hierarchy of the original file. However items will be copied or deleted as indicated by the function.

Control records will be updated and balanced after the file has been created. The item(s) to be edited can be in the form of a CSV list or single items entered via the screen parameters.

List Copy

Copy a list of items into a new file – file based)

The List Copy function will create a new X9.37 file containing *only* the items in a "pick list file".

Note: *For testing purposes, create an ASCII file containing 1,1,1 and name it:*
c:\tempdemotest\picklist.txt

This test file will select the first item in cash letter 1 and bundle 1. (See General information for details on list files.)

All of the original file bundle and cash letter structure is maintained. However only *selected* item(s) will be in the bundles. After being built, the file will be rebalanced.

Example: Create a new X9.37 containing items included in "Pick List file.

Screen Parameters

Working Directory: Working file created prior to starting
c:\tempdemotest

File Name: Input file
c:\tempAX9\temp.nsf.937

Auxiliary File: (list file we created above with the 1,1,1 values)
c:\tempdemotest\picklist.txt

The output file in the Working Directory will have the same name as the original appended with the extension _COPY.937 and contain the one item specified.

c:\tempdemotest\temp.nsf.937_COPY.937 **(View in X9Viewer)**

List Delete

Creates a new file except for those in a list – file based.

The List Delete function will create a new X9.37 file containing all the items in the original *file except* those in the List file.

The new file will be located in the Working Directory and will have the same filename as the source (File Name) appended with the characters "_COPY.937" (e.g. NEWTEMP_COPY.937).

All of the original file bundle and cash letter structure is maintained. However items selected in the list file *will be missing* in the bundles. After being built, the file will be rebalanced.

Note: a Copy file and a Delete file are both appended with _COPY.937. A delete is simply a copy with records removed.

Example: Create new X9.37 except items included in "Pick List file.

Screen Parameters

Working Directory: Working file created prior to starting
c:\tempdemotest

File Name: Input file
c:\tempAX9\temp.nsf.937

Auxiliary File: (list file we created above with the 1,1,1 values)
c:\tempdemotest\picklist.txt

The output file in the Working Directory will have the same name as the original appended with the extension _COPY.937 and will contain the full file minus the item deleted.

c:\tempdemotest\nsf.937_COPY.937 **(View in X9Viewer)**

Item Copy (Copy an Item into a new X9.37 file)

The Item Copy function will create a new X9.37 file containing only the *single i*tem in the source file identified by the screen entry of cash letter, bundle, and item.

Example: Copy Item to new X9.37 file:

Screen Parameters

Working Directory: Working file created prior to starting
c:\tempdemotest

File Name: Input file
c:\tempax9\temp.nsf.937

Auxiliary File: (may be required, but not used)
c:\tempdemotest

Cash Letter: Specifies which existing cash letter contains the item.
1

Bundle: Specifies which existing bundle contains the item.
1

Item: Specifies which existing item to extract.
1

The output file in the Working Directory will have the same name as the original appended with the extension _COPY.937. In the example, it will contain the first item of the first bundle of the first cash letter.

c:\tempdemotest\temp.nsf.937__COPY.937 (View in X9Viewer)

Item Delete

Creates a new X9.37 file except for the single item identified.

The Item Delete function will create a new X9.37 file containing all the items in the original file ***except*** the single item identified in the parameter structure of the screen.

The new file will be located in the Working Directory and will have the same filename as the source File Name field appended with the characters _COPY.937.

All of the original file bundle and cash letter structure is maintained. However items selected in the list file will be missing in the bundles. After being built, the file will be rebalanced.

Note: a Copy file and a Delete file are both appended with _COPY.937. A delete is simply a copy with records removed.

Example: Create new X9.37 except single item identified by screen parameters:

Screen Parameters

Working Directory: Working file created prior to starting
c:\tempdemotest

File Name: Input file
c:\tempax9\temp.nsf.937

Auxiliary File: (may be required, but not used)
c:\tempdemotest

Cash Letter: Specifies which existing cash letter contains the item to delete.
1

Bundle: Specifies which existing bundle contains the item.
1

Item: Specifies which existing item to extract.
1

The output file in the Working Directory will have the same name as the original appended with the extension _COPY.937. It will contain all the items in the file except for the item identified to delete by the screen parameters, which in the example is the first item of Bundle 1 of Cash Letter 1.

c:\tempdemotest\temp.nsf.937__COPY.937 (View in X9Viewer)

Split Functions

This set of functions is concerned with creating new X9.37 files by splitting the original hierarchy.

The resulting output files will only have a single Cash Letter or a single Bundle depending on the function. Control records will be updated and balanced after the file has been created.

Split (Split by Bundle – new file for each Bundle)

The Split function will create a new X9.37 file for every *Bundle* in the original file.

Each resulting file will have one Cash Letter and one Bundle Output files will have the same Byte order and Character set as the original file.

The resulting files will have _BUN_A_B.937 appended to the source File Name field where A is the original file cash letter number and B is the original file bundle number.

Example: Split Bundles to new X9.37 files:

Screen Parameters

Working Directory: Working file created prior to starting
c:\tempdemotest

File Name: Input file to be split
c:\tempax9/temp.nsf.937

The output files in the Working Directory will have BUN_>cashletter>_>Bundle> appended to the filename. The test file only has one bundle so, you will see the following file.

c:\tempdemotest\temp.nsf.937_BUN_1_1.937 (View in X9Viewer)

If there were multiple bundles in the file, you may see the following files.

c:\tempdemotest\temp.nsf.937_BUN_1_2.937

c:\tempdemotest\temp.nsf.937_BUN_1_3.937

Split Cash Letter (New file for each Cash Letter)

The Split Cash Letter function will create a new X9.37 file for each Cash Letter in the original file.

Each resulting file will have one Cash Letter and Output files have the same Byte order and Character set as the original file.

The resulting files will have _ICL_A.937 appended to the source File Name field where A is the original file cash letter number index number.

Example: Split Cash Letters to new X9.37 files:

Screen Parameters

Working Directory: Working file created prior to starting
c:\tempdemotest

File Name: Input file to be split
c:\tempax9/temp.nsf.937

The output files in the Working Directory will have ICL_>index>.937 appended to the filename where >index is the Cash Letter number in the original file.

c:\tempdemotest\temp.nsf.937_ICL_1.937 (cash letter 1) (View in X9Viewer)

If there were multiple Cash Letters in the file, you may see the following files.

c:\tempdemotest\temp.nsf.937_ICL_2.937 (cash letter 2)
c:\tempdemotest\temp.nsf.937_ICL_3.937 (cash letter 3)

Append/Prepend Functions

Each of these Append/Prepend functions inserts new CSV or X9.37 item record(s) into the source X9.37 file.

The result of every function is a fully formed X9.37 file that contains all the hierarchy of the original file with the new items inserted at positions specified by the function type.

For comparison, it is a good idea to save a copy of your original source file prior to executing these functions, because the specified items are added to the original source file in the File Name field.

Several steps need to be completed to create the item file that is to be appended/prepended prior to executing the Append/Prepend function:

Step 1: Determine items you want to append/prepend – Example: Looking for all $6000.00 items.

Create a text file in C:\tempdemotest\findlist.txt
Enter the values:
000600000 to indicate you want to find all the checks in the file for $6,000.00
Execute the steps under **Find Pick 9.37** function.
This will create a text file with the cash letter, bundle, item number(s) that match the given criteria.

Step 2: Select items – SELECTLIST9.37 or SELECTLISTCSV
Follow the example under the section Select List 9.37 or Select List CSV depending upon the file you want to select the items *from.* You are either selecting *from* a 9.37 file or a CSV file.

The file created from this function does not build any headers/trailers – only selects the records from the input file. *This is the file you use in the Auxiliary File field and is an input file of items to be added.*

Step 3: Append/Prepend records to a 937 –
If you are appending from a 937 file, follow procedures Append 9.37.
If you are appending from a CSV file, follow procedures Append CSV.

See General Information - Append/Prepend - EXAMPLE OF Insertion (Page 25) for the table of Inserts displaying the effect on the file of where the Append and Prepend functions place the inserted data.

Append 9.37

Control records will be updated and balanced after the file has been modified.

The Append 9.37 function will change the source X9.37 file by appending items located in a separate X9.37 format file to the Item position of the source file.

The resulting file will have the same number of Cash Letters and Bundles as the source *but with more items.*

Append means that the data will be inserted **after** the item identified by the screen parameters Cash Letter, Bundle, and Item.

The X9.37 data being appended must be in the same format as the source file it is being inserted into.

NOTE: The X9.37 file that has the data to be appended ***must*** have been created by one of the following:

Select List 9.37
Select Item 9.37

Example: Append X9.37 records to a X9.37 file:

Screen Parameters

Working Directory: Working file created prior to starting
c:\tempdemotest

File Name: The output file to which the records will be added.
c:\tempax9\temp.nsf.937

Auxiliary File: This is an input file of items to be added.
c:\ tempdemotest\temp.nsf.937_copy.937

(The above file was created by Select List 9.37 and contains 1 item record to add to the source file. See Select List 9.37)

*Cash Letter: Specifies which existing cash letter **receives** the item.*
1

*Bundle: Specifies which existing bundle **receives** the added items.*
1

*Item: Specifies which existing item to **follow** with the added items.*
2

*Record: Specifies which existing record to **follow** with the added items.*
25

Field: Specifies which existing field to follow with the added items.
(If last field, leave blank)

The appended items should be added to the file designated in File Name field after Record 25 of Item 2, Bundle 1, Cash Letter 1.

c:\tempax9\temp.nsf.937 **(View in X9Viewer)**

Append CSV – Puts CSV items in a X9.37 file

The Append CSV function will change the *source* X9.37 file by appending items in CSV format to the Item position in the source X9.37 file.

It does this by changing the append data CSV file to a temporary X9.37 file and then append the X9.37 items to the Item position specified by the screen parameters into the source X9.37 files.

The resulting file will have the same number of Cash Letters and Bundles as the source but with more items. Append means that the data will be inserted **after** the item identified by the Cash Letter, Bundle, and Item..

NOTE: The X9.37 file that has the data to be appended ***must*** have been created by one of the following:

Select List CSV
Select Item CSV

Example: Append CSV records to a X9.37 file:

Screen Parameters

Working Directory: Working file created prior to starting
c:\tempdemotest

File Name: The output file to which the records will be added.
c:\tempax9\temp.nsf.937

Auxiliary File: This is an input file of items to be added.
c:\ testdemotest\temp.csv.937_copy.937.CSV

(This file was created by Select ListCSV contains item records to be added to the source file.)

*Cash Letter: Specifies which existing cash letter **receives** the item.*
1

*Bundle: Specifies which existing bundle **receives** the added items.*
1

*Item: Specifies which existing item to **follow** with the added items.*
1

*Record: Specifies which existing record to **follow** with the added items.*
25

Field: Specifies which existing field to follow with the added items.
(If last field, leave blank)

The appended items should be added to the file designated in File Name field.

c:\tempax9\temp.nsf.937 **(View in X9Viewer)**

Prepend 9.37

Adds items to the file prior to item position specified by the screen Parameters.

Prepend 9.37 function will change the source X9.37 file by prepending items located in a separate X9.37 format file *before the Item position of the source file.*

The resulting file will have the same number of Cash Letters and Bundles as the source but with more items. Prepend means that the data will be inserted **before** the item identified by the Cash Letter, Bundle, and Item fields on the screen.

The X9.37 data being prepended must be in the same format as the source file it is being inserted into.

NOTE: The X9.37 file that has the data to be prepended ***must*** have been created by one of the following:

Select List 9.37
Select Item 9.37

Example: Prepend X9.37 records to a X9.37 file:

Screen Parameters

Working Directory: Working file created prior to starting
c:\tempdemotest

File Name: The output file to which the records will be added.
c:\tempax9\temp.nsf.937

Auxiliary File: This is an input file of items to be added.
c:\ tempdemotest\temp.nsf.937_copy.937

(This file was created by Select List 9.37 and contains 1 item record to add to the file.)

*Cash Letter: Specifies which existing cash letter **receives** the item.*
1

*Bundle: Specifies which existing bundle **receives** the added items.*
1

*Item: Specifies which existing item **is prior** to the added items.*
1
*Record: Specifies which existing record **is prior** to the added items.*
25
*Field: Specifies which existing field **is prior** to the added items.*
(If last field, leave blank)

The prepended items should be added to the file designated in File Name field.

c:\tempax9\temp.nsf.937

Prepend CSV

Adds CSV items to a X9.37 file prior to item position specified by the screen parameters

The Prepend CSV function will change the *source X9.37* file by prepending item(s) from a CSV format at the Item position of the source file.

Like the function Append CSV, the CSV data will be converted to X9.37 format before insertion. The changed file will have the same number of Cash Letters and Bundles as the source but with more items.

Prepend means that the data will be inserted **before** the item identified by the Cash Letter, Bundle, Item triplet . The X9.37 data being prepended must be in the same format as the file it is being inserted into. This CSV to X9.37 conversion is handled automatically by Prepend CSV.

NOTE: The X9.37 file that has the data to be prepended ***must*** have been created by one of the following:

Select List CSV
Select Item CSV

Example: Prepend CSV records to a X9.37 file:

Screen Parameters

Working Directory: Working file created prior to starting
c:\tempdemotest
File Name: The output file to which the records will be added.
c:\tempax9\temp.nsf.937
Auxiliary File: This is an input file of items to be added.
c:\ testdemotest\temp.csv.937_copy.937.CSV
(This file was created by Select ListCSV contains item records to be added to the source file.)

*Cash Letter: Specifies which existing cash letter **receives** the item.*
1
*Bundle: Specifies which existing bundle **receives** the added items.*
1
*Item: Specifies which existing item is **prior to** the added items.*
1
*Record: Specifies which existing record **is prior** to the added items.*
25
*Field: Specifies which existing field **is prior** to the added items.*
(If last field, leave blank)

The prepended items should be added to the file designated in Source File field.

c:\tempax9\temp.nsf.937 **(View in X9Viewer)**

Balance Functions – Balance the Control Records

This set of functions is concerned with balancing the control record fields. There are two balance functions available with AX9Lib, Test Balance and Balance.

Conditional fields in the source file are not balanced, i.e. Image Count.

A Summary File Report that contains the counts and amounts summed from data found in the item records will be generated in the Working Directory for each of the following Balance functions on the screen. See page 45 for a detailed explanation of the output Summary File. The file name will be File Name.TOC.SUM.

For X9.37 files to be in *balance* it means that the contents of item records (25 and 31) total to the Control Records for bundles, cash letters and the file as a whole. The Control Records are records 70, 90 and 99.

The Control Records do not total the same elements on the different control records. For example, the file control record 99 has a Records Total but no other control record has this type of total. In addition, the control totals may be conditional and in this case there may be no content for a given total. In this case the file is not Out of Balance because the total is not specified.

The fields which appear in one or more Control Records are:

Cash Letter Count (File) -	Record 99
Total Record Count (File) -	Record 99
Bundle Count (Bundle & Cash Letter) -	Record 70 & 90
Item Count (Bundle, Cash Letter, File) -	Record 70, 90 & 99
Amount (Bundle, Cash Letter, File) -	Record 70, 90 & 99
Images Count (Cash Letter, File) -	Record 90, & 99
MICR Valid Amount (Bundle) -	Record 70

Test Balance

Test Balance will verify the source file balances. The source file will be tested for *balan*ce. The return value will be non-zero if the file is not in *balance*. **The file contents will not be change in either case.**

Example: Test Balance

Screen Parameters

File Name: The name of the file that will be verified.
c:\tempax9\temp.nsf.937
Working Directory:
c:\test3
Cash Letter: Specifies which existing cash letter will be verified.
1
Bundle: Specifies which existing bundle will be verified.
1

See Balance Summary File explanation for output.
c:\tempdemotest\temp.nsf.937.TOC.SUM

Balance – Balance File, Cash Letters, or Bundles

The source file will be updated so that it is in *balan*ce. *The return value will be non zero if the file could not be made to balance.*

Example: Balance

Screen Parameters

File Name: The name of the file that will be verified.
c:\tempax9\temp.nsf.937

Working Directory:
c:\test3

Cash Letter: Specifies which existing cash letter will be verified.
1

Bundle: Specifies which existing bundle will be verified.
1

See Balance Summary File explanation for output.t

c:\tempdemotest\temp.nsf.937.TOC.SUM

Balance Summary File – Output files from Balance Screens (not a screen)

Each of the Balance screen functions described above will generate a Summary file into the Working Directory. The TOC.SUM contains the counts and amounts summed from data found in the item records.

The Summary file is CSV format for easy loading. To determine the contents of a line, use the following:

Bundle Control Line--

Cash Letter number and Bundle number are non zero

Cash Letter Control Line--

Cash Letter number is non zero and Bundle number is zero

File Control Line--

Cash Letter number and Bundle number are both zero

The columns are as follows:

Line Number
Cash Letter Number
Bundle Number
Amount
Items
Images(Cash Letter, Bundle)
MICR Valid Amount(Bundle)
Bundles (Cash Letter)
Records(File)
Cash Letters(File)

Example File Name in Working Directory

c:\tempdemotest\temp.nsf.937.TOC.SUM

Sample Summary File

```
1,  1,  1,      2400000,    3,    6,              2400000
2,  1,  0,      2400000,    3,    6,          0,     1
3,  0,  0,      2400000,    3,    0,                    0,     0,     27,     1
```

Field Functions

This set of functions is concerned with reading and writing the individual fields in a X9.37 record down to the Item level. This means that control records and the initial record of each item can be edited.

Get Field

The parameters Cash Letter, Bundle, Item, Record, Occurrence, Field are used to define the location of a field to read. The field data is returned in the Return Code\Data parameter. All standard record types can be interrogated by this function.

When there are multiple identical record numbers in an item (e.g. endorsements), set the Occurrence value to the desired record. Use an Occurrence value of 0 to get a particular field from the first of multiple X9.37 records. In a similar way use an Occurrence value of 100 to point at the last record.

Since the contents of the file are not known, a series of inquires can be made with parameter values of "0" indicating the selection range. The table below shows which records are referenced when the information is not known.

The following examples would be used to search for information.

Get Field (filename,0,0,0,99,2,0,data) will report the number of Cash Letters in the file.
Get Field (filename,1,0,0,90,2,0,data) will report the number of Bundles in the first Cash Letter.
Get Field (filename,1,1,0,70,3,0,data) will report the Amount for the 1st Bundle in the 1st Cash Letter.

Reporting Selection (for parameters cash letter, bundle and item)

Location	Range	Records
0,0,0	All File	01,99
N,0,0	All of Nth cash letter	10,99
N,M,0	All of Mth bundle of Nth cash letter	
20,70		
N,M,P	The pth item of Mth bundle of Nth cash letter	25,31

Note: Get Field can also access records with 187 format by adding 18700 to the record number.

Example 1: Display number of Cash Letters in the file.

Screen Parameters

File Name: The name of the file from which to retrieve the data.
c:\tempax9\temp.nsf.937

Working Directory:
c:\test3

Cash Letter: Specifies which existing cash letter from which to retrieve the data.

0

Bundle: Specifies which existing bundle from which to retrieve the data.

0

Item: Specifies which existing item from which to retrieve the data.

0

Record: Specifies which existing record from which to retrieve the data.

99

Field: Specifies which existing field from which to retrieve the data.

2

Occurrence.

0

The screen will display:

Completed successfully.
Field Data = 000001
Would you like to set the parameter
Yes or No

If you click yes, it will display the value in the Return Code/Data field on the screen.

Return Code/Data:

000001

Example 2: Display the amount of a Detail record in Cash Letter 1, Bundle 1, and Item 1.

Screen Parameters

File Name: The name of the file from which to retrieve the data.

c:\tempax9\temp.nsf.937

Working Directory:

c:\test3

Cash Letter: Specifies which existing cash letter from which to retrieve the data.

1

Bundle: Specifies which existing bundle from which to retrieve the data.

1

Item: Specifies which existing item from which to retrieve the data.

1

Record: Specifies which existing record from which to retrieve the data.

25

Field: Specifies which existing field from which to retrieve the data.

7

The screen will display:

Completed successfully.
Field Data = 0000900000
Would you like to set the parameter
Yes or No
If you click yes, it will display the value in the Return Code/Data field on the screen.

Return Code/Data:

0000900000

Example 3: Access records with 187 format – Add 18700 to the record number and use the Get Field function.

Screen Parameters

File Name: The name of the file where to write the data.
c:\tempax9\temp.nsf.937

Working Directory:
c:\test3

Cash Letter: Specifies which existing cash letter contains the field.
1

Bundle: Specifies which existing bundle contains the field.
1

Item: Specifies which existing item that contains the field.
1

Record: Specifies which existing record that contains the field.
18752 (18700 + 52 record number)

Field: Specifies the field where the data will be written.
2

Occurrence:
0

Set Input Data: The data to be written in the field
987654320

Get Field Optimize

This is an extension of the Get Field function only the data used for this function is retrieved from fields of an X9.37 using an already created TOC file.

The parameters Cash Letter, Bundle, Item, Record, Occurrence, Field are used to define the location of a field to read. The field data is returned in the Return Code\Data parameter. All standard record types can be interrogated by this function.

When there are multiple identical record numbers in an item (e.g. endorsements), set the Occurrence value to the desired record. Use an Occurrence value of 0 to get a particular field from the first of multiple X9.37 records. In a similar way use an Occurrence value of 100 to point at the last record.

Since the contents of the file are not known, a series of inquires can be made with parameter values of "0" indicating the selection range. The table below shows which records are referenced when the information is not known.

The following examples would be used to search for information.

Get Field (filename,0,0,0,99,2,0,data) will report the number of Cash Letters in the file.
Get Field (filename,1,0,0,90,2,0,data) will report the number of Bundles in the first Cash Letter.
Get Field (filename,1,1,0,70,3,0,data) will report the Amount for the 1st Bundle in the 1st Cash Letter.

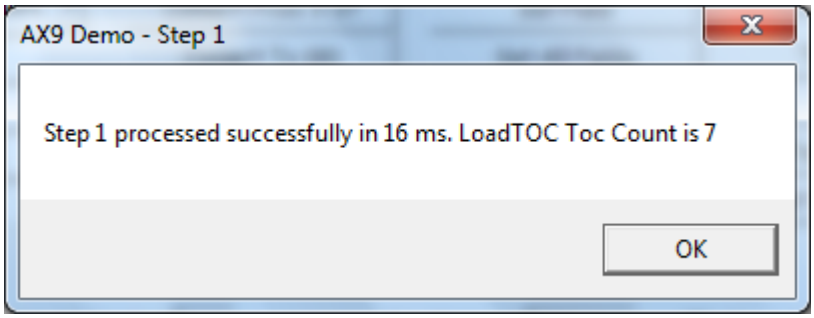
Reporting Selection (for parameters cash letter, bundle and item)

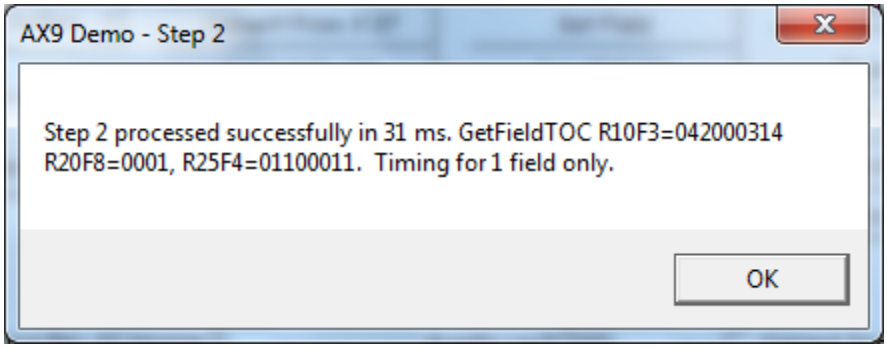
Location	Range	Records
0,0,0	All File	01,99
N,0,0	All of Nth cash letter	10,99
N,M,0	All of Mth bundle of Nth cash letter	
20,70		
N,M,P	The pth item of Mth bundle of Nth cash letter	25,31

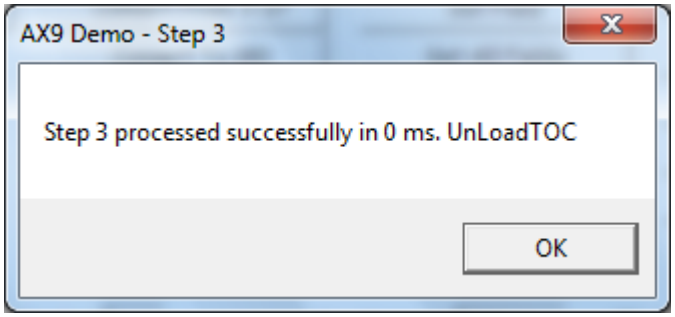
N,0,0	All of Nth cash letter	10,99
N,M,0	All of Mth bundle of Nth cash letter	
20,70		
N,M,P	The pth item of Mth bundle of Nth cash letter	25,31

There are specific steps to complete this function. (See drop down box)
data is retrieved from fields of an X9.37 using an already created TOC file.

Step 1 - Load TOC
Step 2 - Get Field TOC
Step 3 - Unload TOC
Examples of all steps.


AX9 Demo - Step 1
Step 1 processed successfully in 16 ms. LoadTOC Toc Count is 7
OK


AX9 Demo - Step 2
Step 2 processed successfully in 31 ms. GetFieldTOC R10F3=042000314
R20F8=0001, R25F4=01100011. Timing for 1 field only.
OK


AX9 Demo - Step 3
Step 3 processed successfully in 0 ms. UnLoadTOC
OK

Set Field

The parameters Cash Letter, Bundle, Item, Record, Occurrence, and Field are used to define the location of a field to write. The field data to be written is passed in the Set Input Data field. All of the normal record types can be updated by this function (01, 10, 20, 25, 26, 27, 28, 31, 32, 33, 34, 35, 50, 52, 54, 61, 70, 90 and 99**). When setting a Record 61 or Record 68, be sure to set the correct Destination Record61/68 and Source Record61/68 versions**.

When there are multiple identical record numbers in an item (e.g. endorsements), set the Occurrence value to the desired record. Use an Occurrence value of 0 to set a particular field in the first of multiple X9.37 records. In a similar way use an Occurrence value of 100 to point at the last record.

The data written to the field will be converted to the Character Type of the destination file. The data content must be at least the width of the destination field and cannot contain characters less than space (0X20).

Note: If setting Control Record 1 - File Header, use CL, Bundle, Item =0, Record 1, Field 13

Note: If setting Record 52, use CL, Bundle, Item =1 (or whichever), Record 52, Field 2, Occurence 1

Note: Set Field can also access records with 187 format by adding 18700 to the record number.

Example: Set Cash Letter Creation Date

Screen Parameters

File Name: The name of the file where to write the data.
c:\tempax9\temp.nsf.937

Cash Letter: Specifies which existing cash letter contains the field.
1

Bundle: Specifies which existing bundle contains the field.
1

Item: Specifies which existing item that contains the field. (If setting a Control Record must be 0.)
1

Record: Specifies which existing record that contains the field.
20

Field: Specifies the field where the data will be written.
6

Occurrence: If multiple occurrences of record, must be correct occurrence

0

Set Input Data: The data to be written in the field

20060717

The modified field will be updated in Record 20 the file specified in File Name field

c:\tempax9\temp.nsf.937

Example2: Access records with 187 format – Add 18700 to the record number.

Screen Parameters

File Name: The name of the file where to write the data.

c:\tempax9\temp.nsf.937

Cash Letter: Specifies which existing cash letter contains the field.

1

Bundle: Specifies which existing bundle contains the field.

1

Item: Specifies which existing item that contains the field.

1

Record: Specifies which existing record that contains the field.

18752 (18700 + 52 record number)

Field: Specifies the field where the data will be written.

2

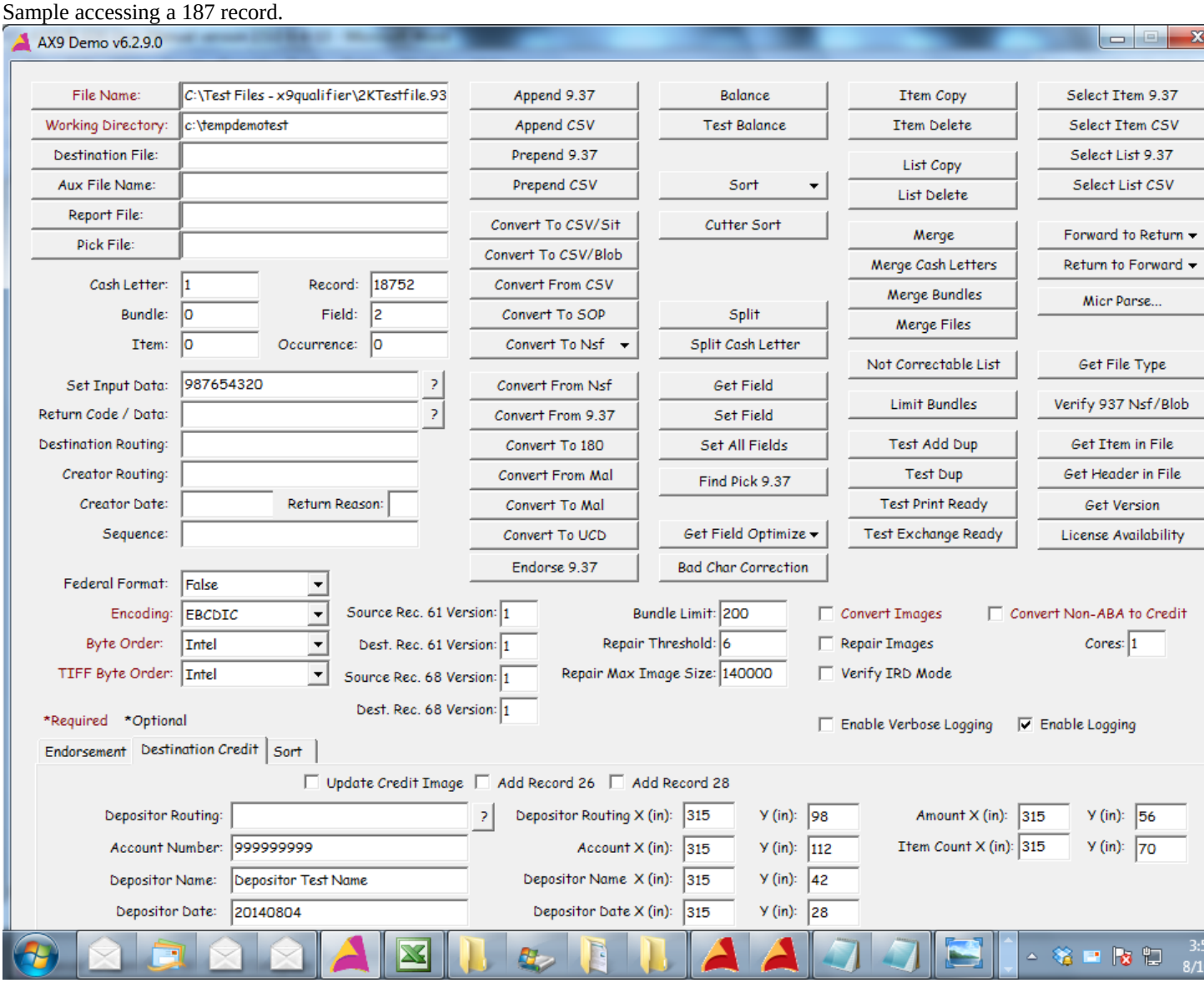
Occurrence:

0

Set Input Data: The data to be written in the field

987654320

Sample accessing a 187 record.



Example3: Access records with Record 61 format

Screen Parameters

File Name: The name of the file where to write the data.
c:\tempax9\temp.nsf.937

Cash Letter: Specifies which existing cash letter contains the field.
1

Bundle: Specifies which existing bundle contains the field.
1

Item: Specifies which existing item that contains the field.
1

Record: Specifies which existing record that contains the field.
61

Field: Specifies the field where the data will be written.
2

Occurrence:
0

Set Input Data: The data to be written in the field
987654320

Source Rec 61 Version = **2 (this indicates the R61 is an AMP2 format)**

Dest Rec 61 Version = **2**

Set All Fields

This function writes data to every occurrence of the specified field in the X9.37 file. The screen parameters Record and Field are used to define the locations of fields *to write*. All occurrences of any record can be written by this function.

The data written to the field will be converted to the Character Type of the destination file. The data content must be at least the width of the destination field and cannot contain characters less than Space (0X20).

Update 1/15/09: Use SET ALL FIELDS to set Record 61 and 84 byte 61 Record. SetAllFields can be used to change fields in the Record 61. Make sure to use the correct AMP Source 61 and Destination 61 Version parameter. E.g.: if Dest Rec = 47

when you created the record, you would use a Source Rec = 2 and a Dest Rec = 2 when you modify. Dest=47 is a specialty format equating to a AMP2 Record 61. (See Appendix F for AMP Record 61 definitions). Dest=47 creates a Record 61 after the 10 record, so the Cash Letter is 1 and 0's for Bundle and Item, Record=61, Field= field you wish to change.

Example 1: Set Cash Letter Business Date

Screen Parameters

File Name: The name of the file where to write the data.
c:\tempax9\temp.nsf.937

Cash Letter: Specifies which existing cash letter contains the field.
0

Bundle: Specifies which existing bundle contains the field.
0

Item: Specifies which existing item that contains the field.
0

Record: Specifies which existing record that contains the field.
10

Field: Specifies the field where the data will be written.
5

Occurrence:
0

Set Input Data: The data to be written in the field
20060717

The modified field will be updated in all occurrences of Record 10 the file specified in File Name field

c:\tempax9\temp.nsf.937

Example 2: Set Posting Account # in all Record 61s

Screen Parameters

File Name: The name of the file where to write the data.
c:\tempax9\temp.nsf.937

Cash Letter: Specifies which existing cash letter contains the field.
0

Bundle: Specifies which existing bundle contains the field.
0

Item: Specifies which existing item that contains the field.
0

Record: Specifies which existing record that contains the field.
61

Field: Specifies the field where the data will be written.
6

Occurrence:
0

Set Input Data: The data to be written in the field
12345678/077

The modified field will be updated in all occurrences of Record 61 the file specified in File Name field

c:\tempax9\temp.nsf.937

Example 3: Set Posting Account # in file where the Record 61 is originally created using a Dest=47 Record 61

Screen Parameters

File Name: The name of the file where to write the data.
c:\tempax9\temp.nsf.937

Cash Letter: Specifies which existing cash letter contains the field.
1

Bundle: Specifies which existing bundle contains the field.
0

Item: Specifies which existing item that contains the field.
0

Record: Specifies which existing record that contains the field.
61

Field: Specifies the field where the data will be written.
6

Occurrence:
0

Set Input Data: The data to be written in the field
12345678/077

Source Rec 61: Specifies which AMP Record 61 you are reading.
2

Dest Rec. 61 Version : Specifies which AMP Record 61 you are writing.
2

The modified field will be updated in all occurrences of Record 61 the file specified in File Name field

c:\tempax9\temp.nsf.937

1.

Select Functions

The Select set of functions is concerned with selecting the records for subsequent processing.

The resulting file can be in either a CSV or X9.37 format. The resulting file is ***no***t a fully formed X9.37 or CSV equivalent of a X9.37 and will only contain the records for the items selected.

The resulting temp file will contain *only the items selected* in either X9.37 or CSV format. In the case of CSV, the 937 is generated first and then converted to CSV format.

Select List 9.37

The Select List 9.37 function will select and copy items from an existing X9.37 file to a temporary output file based on a specified list (Pick List file).

Example: Copy selected items

Screen Parameters

Working Directory: Working file created prior to starting

c:\tempdemotest

File Name: Input file

c:\tempAX9\temp.nsf.937

Auxiliary File: Input List file of items to copy to new file (see General information for how to create the list)

c**:\tempdemotest\temp.nsf.937 Find.LST**
(This file was created from Find Pick 9.37 and contains the values:
1,1,1

The output files in the Working Directory will have the same filename as the source appended with the characters _COPY.937.

c:\tempdemotest\temp.nsf.937_COPY.937

Note: This function does not build any headers/trailers – only selects the records. (Cannot be viewed in X9 Viewer – parse error, but the file can be used as input to Append/Prepend the records to another file.)

Select List CSV

The Select List CSV function will copy a portion of an *existing CSV* file to a temporary Output file based on a Pick List file.

The resulting file will contain only the items selected in CSV format. During this function, the X9.37 data is generated first and then converted to CSV format.

Note: Convert the CSV to 937 first and then use function Select List CSV to create the CSV list. (10/2007)

Example: Copy selected items

Screen Parameters

Working Directory: Working file created prior to starting
c:\tempdemotest

File Name: Input file
c:\tempAX9\temp.csv.937

Auxiliary File: Input List file of items to copy to new file (see General information for how to create the list)

c**:\tempdemotest\temp.nsf.937 Find.LST**
(this file was created from Find Pick 9.37 and contains the values:
1,1,1

The output files in the Working Directory will have the same filename as the source appended with the characters _COPY.937.

c:\tempdemotest\temp.csv.937_COPY.937.CSV

Note: This function does not build any headers/trailers – only selects the records. (Cannot be viewed in X9 Viewer – parse error, but the file can be used as input to Append/Prepend the records to another file.)

Select Item 9.37

The Select Item 9.37 function will copy a portion of an existing 937 file to a Working Directory file based on a single item.

The resulting temp file will be the records from a single items in X9.37 format and will have the name as the source File Name field with _COPY appended.

Example: Copy Item to new X9.37 file:

Screen Parameters

Working Directory: Working file created prior to starting
c:\tempdemotest

File Name: Input file
c:\tempax9\temp.nsf.937

Cash Letter: Specifies which existing cash letter contains the item.
1

Bundle: Specifies which existing bundle contains the item.
1

Item: Specifies which existing item to extract.
1

The output file in the Working Directory will have the same name as the original appended with the extension _COPY

c:\tempdemotest\temp.nsf.937_COPY.937

Note: This function does not build any headers/trailers – only selects the records. (Cannot be viewed in X9 Viewer – parse error, but the file can be used as input to Append/Prepend the records to another file.)

Select Item CSV

The Select Item CSV function will copy a portion of an existing ***X9.37 file*** to a temporary file based on a single item pick from the parameter block.

The resulting temp file will be **CSV.** In the case of CSV, the X9.37 is generated first and converted to CSV format.

Example: Copy Item to a temporary CSV file:

Screen Parameters

Working Directory: Working file created prior to starting
c:\tempdemotest

File Name: Input file
c:\tempax9\temp.csv.937

Cash Letter: Specifies which existing cash letter contains the item.
1

Bundle: Specifies which existing bundle contains the item.
1

Item: Specifies which existing item to extract.
2

The output file in the Working Directory will have the same name as the original appended with the extension _COPY

c:\tempdemotest\temp.csv.937_COPY

Note: This function does not build any headers/trailers – only selects the records. (Cannot be viewed in X9 Viewer – parse error, but the file can be used as input to Append/Prepend the records to another file.)

Find Pick 9.37

The Find Pick function creates a Pick List in the working directory using a series of values from the file specified by Auxiliary File.

Each value is scanned against the X9.37 file File Name field and if a match is found, the cash letter, bundle and item number(s) associated with that record is saved in the Pick List.

This series of values is located in a Find List file specified by the Auxiliary File on the screen.

Example:

Screen Parameters

Working Directory: Working file created prior to starting
c:\tempdemotest

File Name: Input file
c:\tempax9\temp.nsf.937

Auxiliary File: Input file
c:\tempdemotest\findlist.txt (this file contains 000600000 - Looking for the amount value of $6,000.00)

Cash Letter: Specifies which existing cash letter contains the item.
0

Bundle: Specifies which existing bundle contains the item.
0

Item: Specifies which existing item to extract.
0

Record: Specifies the record containing the field to match.
25 (this is the item record)
Field: Specifies the field containing the data to match.
7 (this is the item amount)
Occurrence:
0

The new file will be located Working Directory and will have the same filename as the source (File Name) appended with the characters Find.Lst

c:\tempdemotest\temp.nsf.937 Find.Lst (View in Notepad)

The output physical file contains the cash letter, bundle, item number(s) that match the criteria above. In the above example, it would have the following value, meaning it found one record matching the above criteria in each of these cash letters, bundles, and items. This output "Pick List" can then be used as input to various functions of AX9Lib.

1,1,3

Sort/Merge Functions (Routing)

Multiple X9.37 files can be sorted by ***Payor Routing number (Record 25, field 4)*** and then merged so that only items for a single Payor appear in a single file using the Sort Function. This is commonly used to support Forward Files. Used in combination with Merge, multiple cash letters can be sorted and then merged so that there is a resulting file for each Payor bank.

Multiple X9.37 files can be sorted by ***Bank of First Deposit (BOFD – Record 26, Field 3)*** and then merged so that only items for a single BOFD appear in a single file using the SortBofD Function. This is commonly used to support Return Files. Used in combination with Merge, multiple return cash letters can be sorted and then merged so that there is a resulting file for each BOFD bank. For this function, an Auxiliary file can be input to SortBofD which contains a list of Return Processors for a given BOFD (Notepad list). This list can be long (thousands of banks). It can be maintained by the banks themselves as they receive updated list or add new exchange partners. Typically, the last Processor in the list is the Federal Reserve and anything not identified is sent to them. The result is a set of files created for the Return Processor and not for the BOFD.

After merging, the resulting files are Balanced and generic header records are added to provide a complete X9.37 file. The user will need to fill in a few file specific items such as Immediate Destination Routing number. These fields can be filled using the AX9Lib Set Field function.

Each resulting file will be placed in the Working Directory and it will have the name of the Routing number with an .937 extension. A TOF(Table of Files) file will be generated in the Working Directory with a list of all the X9.37 files generated.

The Header and Trailer records will be of the same Character type and Byte order as the source file. It is up to the user to make sure that all the source files are of the same Character type and Byte order.

Note: To generate valid X9.37 output, both a Sort and a Merge must be performed. This is true even if only one file is to be sorted.

Sort

The Sort function will sort all Check Items in the File Name (input file) and generate multiple new intermediate files in the Working Directory.

Each new file will be named according to the Check Item **Routing** value and will only contain items that contain the same Routing value.

Each call to Sort requires a unique index code (0 to N, N < 937) for the file being sorted. This index is used to generate an extension for all the sorted records. The index is called the File Set Index on the screen.

IMPORTANT: ***The first call to Sort must have an index code of zero to reset the list of files used in the subsequent Merge.***

Intermediate files do not have X9.37 headers and cannot be viewed using the X9.37Viewer.

The Sort function also generates an output file called AMP.TOF. This file contains a list of all the files that have been created during the sort operation. The Merge function uses this list to compose the merged output.

Example:

Screen Parameters

Working Directory: Working file created prior to starting

c:\tempdemotest

File Name: Input file

c:\tempax9\temp.nsf.937

File Set Index: Specifies the record containing the field to match.

0

Output file in Working Direcotry "Route.inx, where Route is the Routing code (011000112) and inx is the File Set Index value (000):

C:\tempdemotest\AMP.TOF

Contents of file:

011000112.000

987654322.000

SortItems

The SortItems function will sort all Check Items in the File Name input file by record 25 or by record 31 for numeric fields only and generate multiple new intermediate files in the Working Directory.

SortBoFD

Used in conjunction with Merge to group items by Bofd Routing (Bank of First Deposit, Record 26, Field 3). Creates an intermediate file for each Bofd Routing number in the working directory. Also creates AMP.TOF containing a list of these intermediate files.

The SortBoFD function will sort all Check Items in the File Name input file and generate multiple new intermediate files in the Working Directory. This function is similar to the Sort function, **only normally will be used on Return files as the sort is by Bank of First Deposit (BOFD - Record 26, Field 3 in forward presentment and Record 32, Field 3 in a Return File) and not sorted by Payor Bank.**

Each new file will be named according to the Check Item Routing value and will only contain items that contain the same Routing value.

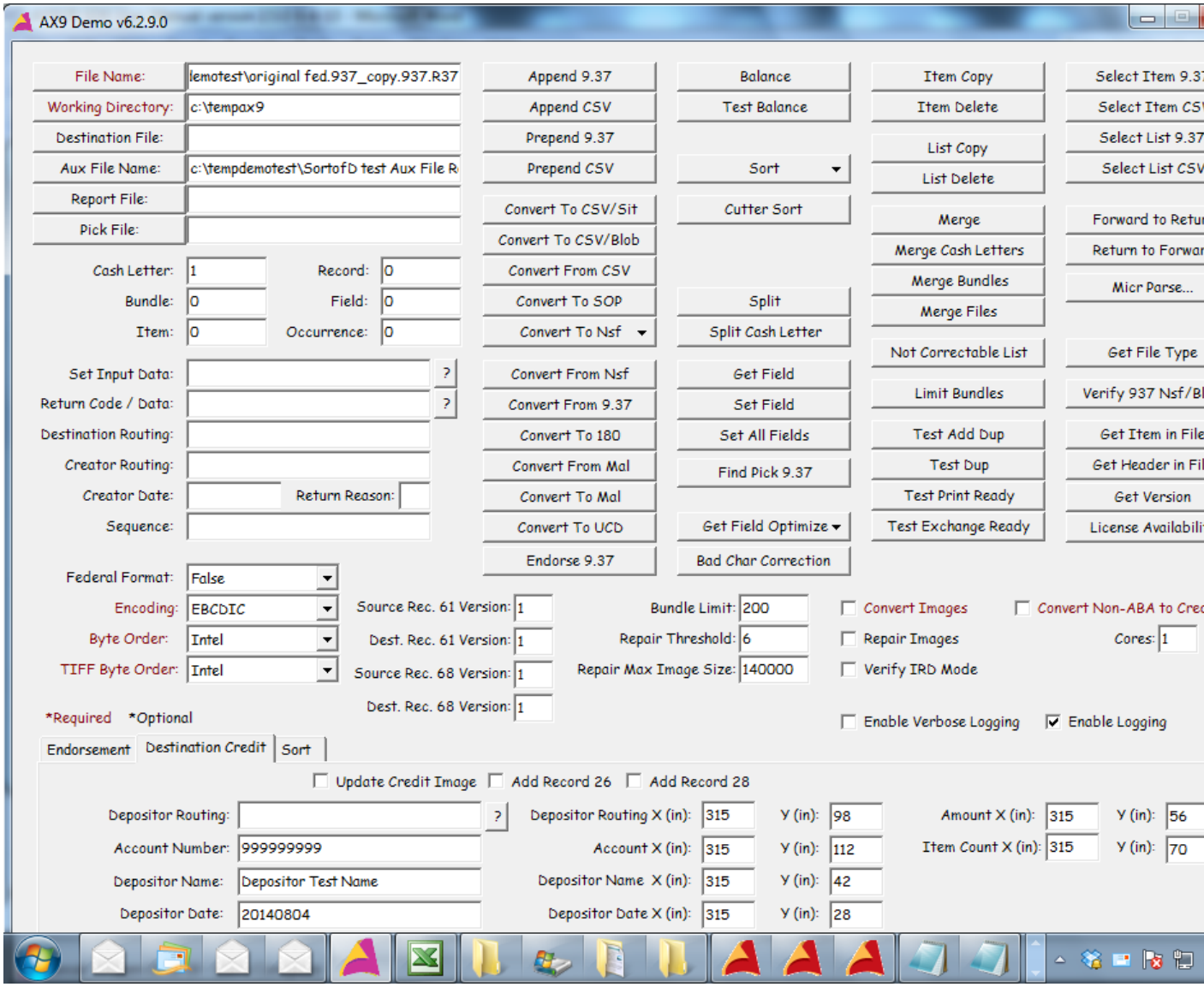
Each call to SortBoFD requires a unique index code (0 to N, N < 937) for the file being sorted. This index is used to generate an extension for all the sorted records. The index is called the File Set Index on the screen.

IMPORTANT: ***The first call to SortBoFD must have an index code of zero to reset the list of files used in the subsequent Merge.***

Intermediate files do not have X9.37 headers, and cannot be viewed in X9Viewer.

The SortBoFD function also generates an output file called AMP.TOF. This file contains a list of all the files that have been created during the sort operation. The Merge function uses this list to compose the merged output.

Example:



Screen Parameters

Working Directory: Working file created prior to starting

c:\tempdemotest

File Name: Input file

c:\tempdemotest\original fed.937_COPY.937.R37 (Return File)

Auxillary File: Input file(file containing a list of Return Processors for a given BOFD)

c:\tempdemotest\SortofD test Aux File Return Processors.txt

File Set Index: Specifies the record containing the field to match.

0

Output files in Working Directory "Route.inx, where Route is the Routing code(011000112) and inx is the File Set Index value (000):

Files:
011000112.000
987654322.000

C:\tempdemotest\AMP.TOF (use this file as input to Merge function)

Contents of AMP.TOF file:

011000112.000

987654322.000

SortBoFD Account

Used in conjunction with Merge to group items by Bofd Account (**Record 26, Field 6).** Creates an intermediate file for each Bofd Account in the working directory. Also creates AMP.TOF containing a list of these intermediate files.

The SortBoFD Account function will sort all Check Items in the File Name input file and generate multiple new intermediate files in the Working Directory. This function is similar to the Sort function, **only isused to sort Bank of First Deposit Account Number (BOFD - Record 26, Field 6) in forward presentment.**

Each new file will be named according to the BOFD Account Number and will only contain items that contain the same Account Number.

Each call to SortBoFD Account function requires a unique index code (0 to N, N < 937) for the file being sorted. This index is used to generate an extension for all the sorted records. The index is called the File Set Index on the screen.

IMPORTANT: ***The first call to SortBoFDAccount must have an index code of zero to reset the list of files used in the subsequent Merge.***

Intermediate files do not have X9.37 headers, and cannot be viewed in X9Viewer.

The SortBoFD Account function also generates an output file called AMP.TOF. This file contains a list of all the files that have been created during the sort operation. The Merge function uses this list to compose the merged output.

Screen Parameters

Working Directory: Working file created prior to starting
c:\tempdemotest

File Name: Input file
c:\test3\temp.nsf.937

File Set Index: Specifies the record containing the field to match.
0

Output files in Working Directory "xxxx.0000, where xxxx is the Account number(123456780) and inx is the File Set Index value (000):

Files:
123456780.000
987654322.000

C:\tempdemotest\AMP.TOF (use this file as input to Merge function)

Contents of AMP.TOF file:

011000112.000

987654322.000

Sort Payor

Same as Sort except can pass in Auxiliary file. Used in conjunction with Merge to group items by Payor Routing **(Record 25, Field 4).** Creates an intermediate file for each Payor Bank Routing Number in the working directory. Also creates AMP.TOF containing a list of these intermediate files.

The Sort Payor function will sort all Check Items in the File Name (input file) and generate multiple new intermediate files in the Working Directory. This function is similar to the Sort function, **only using Record 25, F4 as the sort of the Payor Bank Routing Number.**

Each new file will be named according to the Payor Routing Number and will only contain items that contain the same Routing value.

Each call to Sort Payor requires a unique index code (0 to N, N < 937) for the file being sorted. This index is used to generate an extension for all the sorted records. The index is called the File Set Index on the screen.

IMPORTANT: ***The first call to Sort Payor must have an index code of zero to reset the list of files used in the subsequent Merge.***

Intermediate files do not have X9.37 headers, and cannot be viewed in X9Viewer.

The Sort Payor function also generates an output file called AMP.TOF. This file contains a list of all the files that have been created during the sort operation. The Merge function uses this list to compose the merged output.

Example:

Screen Parameters

Working Directory: Working file created prior to starting
c:\tempdemotest

File Name: Input file
c:\test3\temp.nsf.937

AuxFile Name: Optional. File containing a list of Return Processors for a given Payor

File Set Index: Specifies the record containing the field to match.
0

Output files in Working Directory "xxxx.0000, where xxxx is the Payor Routing Number(01100011) and inx is the File Set Index value (000):

Files:
01100011.000
02100011.000

C:\tempdemotest\AMP.TOF (use this file as input to Merge function)

Contents of AMP.TOF file:

01100011.000

02000011.000

Group Account

The Group Account function will extract all items and group them into multiple files based on the On-Us account value located in **record 25, field 6** of the X9.37 file.

This method will sort all check items in the Input file and generate multiple new intermediate files in the Working Directory. Each new file will be named according to the check item On-Us account value (record 25 field 6) and will only contain items of that same account value. Each call to Group Account requires a unique index code (0 to N, N < 937) for the file being grouped. This index is used to generate an extension for all the sorted records.

The most common use of Group Account is to generate the individual return files for X9.37 Returns. It is common for the return items to go to a Return Processor so the Group Account method provides a Return Processor file list in the Aux File input parameter. When present, this file contains the Return Processor Routing number for each Payor Routing number in the list. The last entry in the list is the Return Processor for all of the account values that are not in the list.

The list would look like the following example:

> Big Bank 1 RT, Big Bank 1 RT
> rt1
> rt2
> rt3
> Big Bank 2 RT, Big Bank21 RT
> rt4
> rt5
> FRB RT, FRB RT

The first route code on a line is assigned to the second route code for return processing. Clearly Big Bank 1 would get their own. All routing codes without a specific second entry, would be assigned to the last Return processor route code. A new pair starts a new Return Processor. The last Return Processor is used for all codes not specifically in the list. In this case, the Fed.

At the conclusion of the function, there would be at most three files with the names of Big Bank 1 RT, Big Bank 2 RT and FRB RT.

IMPORTANT: The first call to Group Account must have an index code of zero to reset the list of files used in the subsequent Merge. **Intermediate files do not have X9.37 headers. but can still be viewed using the Record Tab in the All My Papers X9.37 Viewer.**

The Group Account function also generate an output file called AMP.TOF. This file contains a list of all the files that have been created during the sort operation. The Merge function uses this list to combine all the fields of the same account number into a 937. It uses the TOF in Working Directory from the Group call.

Files Created in the Working Directory:

"Route.inx" where Route is a Routing code and inx is the nFileSet value

AMP.TOF List of all new files created

Screen Parameters

Working Directory: Working file created prior to starting

c:\tempdemotest

File Name: Input file - the name of the source X9.37 file

c:\tempdemotest\original fed.937_COPY.937.R37 (Return File)

Auxillary File: Input file(file containing the routing table)

c:\tempdemotest\SortofD test Aux File Return Processors.txt

File Set Index: Used as the extension for the created files.

0

Sort/Detect Dups

The Sort/Detect Dups function will perform a sort operation on an X9.37 file and identify duplicate items in the file. It is a quick way to determine if a check has been submitted multiple times.

The Sort/Detect Dups function will sort all the checks in the file looking for duplicate items. It is quite fast because there is no real duplicate file generated. **If a destination file is specified, a new X9.37 file containing the duplicate items will be created**. The PickFile property is used as an input file that specifies a pick file where the duplicate items will be listed when the function completes.
If no PickFile is specified, just the duplicate count will be returned on the screen.

Sort/Detect Dups is a quick way to determine if a check has been submitted multiple times. When used with a Merge command, this can be used to perform duplicate detection on a multiple day range of input.

Screen Parameters

Working Directory: Working file created prior to starting

c:\tempax9

File Name: Input file

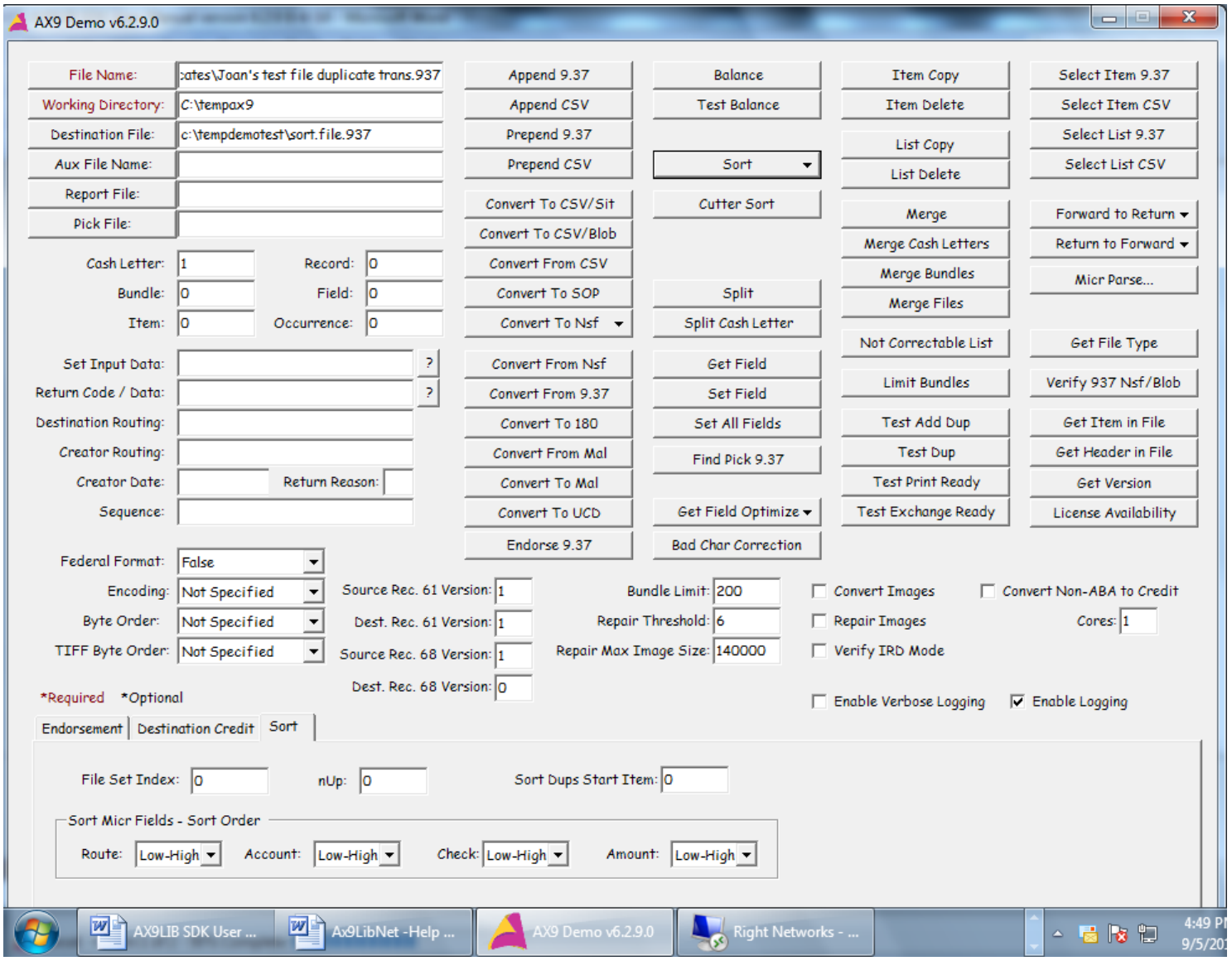
c: C:\Test Files - Duplicates\Joan's test file duplicate trans.937

Destination File: Output 937 file containing only the duplicate items

c:\tempdemotest\Sort File.937 (View in X9Viewer)

Pick File: Input file specifies a pick file where the duplicate items will be listed when the function completes. If no PickFile, just the count will be returned.

c:\tempdemotest\Sort B.txt (A list of items)



Destination File: Output 937 file containing only the duplicate items c:\tempdemotest\Sort File.937 (View in X9Viewer) Pick File: No pick file listed so the count will be returned on the screen.

Sort Dups (No Amount)

The Sort/Detect Dups (No Amount) function will perform a sort operation on an X9.37 file and identify duplicate items in the file, but will not use the Amount field as criteria, only sorts by Payor Routing, Account Number and Check Number.

Screen Parameters

Working Directory: Working file created prior to starting

c:\tempax9

File Name: Input file

c:\temp3\temp DUP TEXT.TXT.937

Destination File: Output 937 file containing only the duplicate items

c:\tempdemotest\Sort File.937 (View in X9Viewer)

Pick File: Input file specified a pick file where the duplicate items will be listed when the function completes. If no PickFile, just the count will be returned.

c:\tempdemotest\Sort B.txt (A list of items)

If there are duplicates found in the file, the return value will be 143 (File Dup found).

Sort Micr Fields

The **Sort Micr Fields** function will perform a sort operation on an X9.37 file and sort all check items in the input file and output a single new file into the Destination file, if specified, or into the working directory if the Destination File is not present. It is often referred to as a Statement Sort.

The Sort Micr Fields function will sort based on the MICR line field data found in each item of the X9.37 file. The sort takes place in field order and sorts by Payor Routing, Account Number, Check Number, and finally Amount.

The sort order parameters specify whether or not to sort by the given parameter, and if so, in which order to sort. A value of -1 indicates that the value should be sorted from lowest to highest, a value of 1 indicates that the value should be sorted from highest to lowest, and a value of 0 indicates that the value should not be sorted.

This function contains a syntax of many Account Number/Check Number formats. The user can supersede the internal formats by adding their own. As an example, the Onus value of 123-456/789 would be represented syntactically as 1-2/3 where the 1,2 and 3 are the numeric fields. In this case the Account number would be the concatenated fields 12 and the

Check number would be the field 3. If the operator needed the same content to parse differently, they could enter 1,23 to get an account of field 1 and a check number of 23.

The inputData field can have as many triplets as needed followed by a triplet of 0,0,0 to indicate completion. The inputData value for the above example would be:

"123-456/789,1,23,0,0,0"

The remaining characters are all treated as part of a numeric field, particularly Blank and Asterisk. Some banks use blanks to separate Account Number and Amount Number strings. This may be problematic because the X9.37 format does not require maintenance of blanks in the MICR line input.

Screen Parameters

Working Directory: Working file created prior to starting

c:\tempax9

File Name: Input source file

c:\tempdemotest\temp.nsf.937

Destination File: Output file file containing only the duplicate items

c:\tempdemotest\Sortdup test. 937

Pick File: Input file specified a pick file where the duplicate items will be listed when the function completes. If no PickFile, just the count will be returned.

c:\tempdemotest\Sortdup PickFile. txt

Sort Micr Fields Sort Order

Route:

Low-High - enables Route sorting from lowest to highest
Off - disables Route sorting
High-Low - enables Route sorting from highest to lowest

Account:

Low-High - enables Route sorting from lowest to highest
Off - disables Route sorting
High-Low - enables Route sorting from highest to lowest

Check:

Low-High - enables Route sorting from lowest to highest
Off - disables Route sorting
High-Low - enables Route sorting from highest to lowest

Amount:

Low-High - enables Route sorting from lowest to highest
Off - disables Route sorting
High-Low - enables Route sorting from highest to lowest

Set Input Data - A string specifying a custom Account Number, Check Number format
triplet values - see above explanation of how to use

Cutter Sort

The Cutter Sort function is used to sort X9.37 files in preparation for IRD Printing. The *File Name* is sorted so that the output can be placed on a multiple page cutter and all the pages cut at once.

The number of cuts is determined by the Up parameter. In order that the resulting stacks of IRDs can be bunched together, multiple pages will have VOID checks in inserted as needed.

Example:

Screen Parameters

Working Directory: Working file created prior to starting

c:\tempdemotest

File Name: Input file

c:\tempax9\temp.nsf.937

Up: Number of cuts

2

Other Merge Functions

There various other Merge and Bundling Functions available with AX9Lib. The following describes the use of these functions and creating new X9.37 files with the specified criteria.

The Destination File is a new parameter to allow the use of a specified output file for the Merge Functions.

Merge

The File Name is a list of sorted files typically created by the Sort function. This list is typically named AMP.TOF and is used as input to the Merge function.

The Merge function will combine all the cash letters within the list of files of the same name (but different extension) into a single file with proper header and trailer records. The file is then Balanced.

Note: In general, any function that reads a 937 file requires source61 and source68 (includes Merge functions), if there are 61 and/or 68 records in the file. In most cases for non-Wells formats, however, source61 and souce61 can use the default 1.

Example:

Screen Parameters

Working Directory: Working file created prior to starting
c:\tempdemotest

File Name: Input file
c:\tempdemotest\AMP.TOF

Source Rec 61: Specifies which AMP Record 61 you are reading
1

Can let default to 1if not present.
Dest Rec. 61 Version : Specifies which AMP Record 61 you are writing.
1
Source Rec 68: Specifies which AMP Record 61 you are reading
1

Dest Rec. 68 Version : Specifies which AMP Record 61 you are writing.
1

Output files in working Directory "Route.937, where Route is the Routing code. All new files created:
C:\tempdemotest\011000112.937
C:\tempdemotest\987654322.937

A list of the files and processing done to each will be in output file:
C:\tempdemotest\AMP.TOF.Log

A list of all new files created:
C:\tempdemotest\AMP.TOF.List or

Merge Cash Letters

The Merge Cash Letter function will combine all the cash letters within a file into a single cash letter with multiple bundles. Each Cash Letter within the original file becomes a new bundle in the new cash letter file.

It will create the proper header and trailer records and balance the file. The *File Name* is the file with the cash letters to be merged.

The resulting file will be placed in the Destination File if it is used, otherwise it will be placed in the Working Directory using the File Name name.

Example:

Screen Parameters

Working Directory: Working file created prior to starting
c:\tempdemotest

File Name: Input file
c:\temp3\My Merge File.937

Destination Source File: Output file
c:\temp3\My Merge Cash Letters.937

Output files in working Directory OR Destination File (these are now fully formatted 937 and can be viewed in X9Viewer):

C:\tempdemotest\My Merge File.937

C:\temp3\My Merge Cash Letters.937

Merge Bundles

The Merge Bundles function will combine all the bundles in each cash letter within a single file into one bundle per cash.. It will create the proper header and trailer records and balance the file.

The resulting file will be placed in the Destination File if it is used, otherwise it will be placed in the Working Directory using the File Name name as the extension.

Example:

Screen Parameters

Working Directory: Working file created prior to starting
c:\tempdemotest

File Name: Input file
c:\temp3\My Merge Cash Letters.937

Destination Source File: Output file (if designated)

c:\temp3\My Merge Bundles.937

Output files in working Directory OR Destination File:

C:\tempdemotest\My Merge Cash Letters.937 OR

C:\temp3\My Merge Bundles.937

Merge Files

The Merge Files function will combine all files contained in a list and the list name entered into the File Name name parameter. Each file will be entered into the new file as a separate Cash Letter with the proper header and trailer records and balanced just as they are.

Example:

Screen Parameters

Working Directory: Working file created prior to starting

c:\tempdemotest

File Name: Input file

c:\tempdemotest\Merge Files Test.txt

(Use Notepad to create a text list of files – complete Path Name)

Destination File: Output file – conditional – if the Destination File parameter Is not used, the resulting file will be placed in the Working Directory

c:\temp3\My Merge File.937

Source Rec 61: Specifies which AMP Record 61 you are reading (if specialty formats are being merge may need different values, eg.=10 BOFA

1

Dest Rec. 61 Version : Specifies which AMP Record 61 you are writing. (if specialty formats are being merge may need different values, eg.=10 BOFA

1

Source Rec 68: Specifies which AMP Record 61 you are reading (if specialty formats are being merge may need different values, eg.=10 BOFA

1

Dest Rec. 68 Version : Specifies which AMP Record 61 you are writing. (if specialty formats are being merge may need different values, eg.=10 BOFA

1

Output files in working Directory with the File Name name and the addition of the .937 extension OR if the Destination File is specified, it will be placed in the Destination File:

C:\tempdemotest\Merge Files Test.937 OR

C:\temp3\My Merge File.937

Limit Bundles

The Limit Bundles parameter allows the setting of the number of items within a bundle. At the bottom of the screen set the "Bundle Limit" parameter to the number of items needed in a bundle.

The Limit Bundle function will read each Cash Letter in the file and limit the number of items in each bundle, create new bundles as necessary, and create the proper header and trailer records and balance the file.

Example:

Screen Parameters

Working Directory: Working file created prior to starting

c:\tempdemotest

File Name: Input file

c:\temp3\My Merge File.937

Destination File: Output file – conditional

c:\temp3\My Limit Bundle File.937

Creator Routing Number ***(if don't fill spaces out Rec 1, Field 5 - 9 digits)***

123456780

Bundle Limit – set the number of items in a Bundle

1 (for test purposes will create 1 item in each bundle)

Output files in working Directory if Destination File not specified.

C:\tempdemotest\My Merge File.937 (Working Directory) OR
C:\temp3\My Limit Bundle File.937 (if Destination File specified)

Return Functions

This set of functions performs two separate functions:

- Creates new X9.37 **Return** formatted files from Forward items
  Note: Use the **Auxiliary File parameter** if using a pick list
- Creates new X9.37 **Forward** items from a Return item
  Note: Use the **Pick File parameter** if using a pick list

Within the X9.37 structure, return items contain most of the same data as the forward check item. However, the location of data on a record and the number of records comprising a return is different from a forward

Using the Return Functions, a single check or a PickList of checks can be converted to a Return file and vice versa, a single Return item or PickList of Return items can create a new Forward item(s).

The reason for a return can be part of the PickList or can be added later with the Set Field Function.

The Return Function not only reorders and reformats a forward record and creates a Return item, but it will also create an endorsement record based on the Creator Routing number in the parameter list. In addition other return records may be inserted based on what is absent in the input Forward X9.37 file.

When **the Federal Format** option is indicated (see page 22 for definition), the system automatically places default values in certain fields if the values are not available on the file:

R32, F2 = '1'
R32, F9 = 'N" (Truncation Indicator - if no record 26)
R33 = record inserted
R33, F4 = ECE item sequence number from R52, F5
R35, F6 (last occurrence) = 'N'
R33, F5 = "yyyymmdd" - current date"
R35, F7 (last occurrence) = '7'
R35, F8 (last occurrence) = '0'
R 35, F5 (last occurrence) = ECE item sequence number from R52, F5

The creation of a Forward item from a Return item reorders and reformats a Return record into a Forward item and creates two Record 28s. The first is created from the original Return record 35 and its return code. The second is a new endorsement record for this representment. The ECE values must be filled from the parameter. They cannot be the values in the Return file.

Finally the counts and amounts are updated.

Forward to Return Functions

Write Return ALL (new 9/09)

The Write Return All function creates a X9.37 file with all the items from the Input File. The Return code is in the Data parameter for all items in the file.

This function will generate returns that conform to ECCHO/i3g recommendations and outputs and **a pick list** (.**pic) of items without record 26s.**

All return generation functions now generate a Destination and/or Pick file if parameter present. Previous results were only in the Working Directory.

When no record 26, the nOccurrence parameter when set to 99 will use the most recent record 28 as the return routing number. If any other value, then the first record 28 will be used.

The file would be rebalanced as a completely self contained Return X9.37 file.

The Write Return All function generates a _COPY version of the input items in the Working Directory or Destination file if present. This file is then converted into a return file in the Working Directory with an extension of R.37.

Example:

Screen Parameters

Working Directory: Working directory file created prior to starting
c:\tempdemotest

File Name: The name of the file from which to retrieve the data.
c:\tempax9\temp.nsf.937

Destination File: The name of the file from which to place the Returns
c:\temp3\temp.nsf.937.R37

Pick File: The name of the file for items with no Record 26's

Creator Routing – Routing number of the creating bank – ***must fill these fields(Error=47)***
987654320

Creator Date – Date return created- ***must fill these fields (Error=47)***
20060622

Sequence- This number is used as the Endorsing Bank Item and will display In field 5 of record type 35.
1

The new Return file will be located in the Working Directory and will have the same filename as the source appended with the extension _COPY.937.R37 or in the Destination File name if designated .

c:\tempdemotest\temp.nsf..937_COPY.937.R37 (View in X9Viewer)

Write Return (from list of Forward Items)

The Write Return function creates a X9.37 file with the items selected from the Pick List specified by Auxiliary File name. (Use triplet and Return Code in Pick List)

Each item is reformatted to that required for a Return Item. If the Pick List contains the single byte Return Reason code (field 6 of record 31), it is added into the Return file. The file maintains the CL and Bundle structure information of the source file *but only contains the items in the PickList.*

The file would be rebalanced as a completely self contained X9.37 file. The numeric value of the Sequence field in the input screen parameters is used as the Endorsing Bank Item Sequence Number in field 5 of the second record 35.

The Write Return function generates a _COPY version of the items specified the Pick List into the Working Directory. This file is then converted into a return file in the Working Directory with an extension of R.37.

Example:

Screen Parameters

Working Directory: Working directory file created prior to starting

c:\tempdemotest

File Name: The name of the file from which to retrieve the data.

c:\tempax9\temp.nsf.937

Auxiliary File: The name of the file containing the list of Return Items

c:\tempdemotest\temp,nsf.937 Find.Lst

(this is the output file of the Find Pick 9.37 function we created containing the cash letter, bundle, item number of a $6,000.00 item – see page 59 for creation of this file. Edit this file and add an A after the item, so you would have 1,1,3,A – this will be the code for the Return)

Creator Routing – Routing number of the creating bank

987654320

Creator Date – Date return created

20060622

Sequence- This number is used as the Endorsing Bank Item and will display In field 5 of record type 35.

1

The new Return file will be located in the Working Directory and will have the same filename as the source appended with the extension _COPY.937.R37.

c:\tempdemotest\temp.nsf..937_COPY.937.R37 (View in X9Viewer)

Write Return Item – Writes a single Return Item

The Write Return Item function creates a X9.37 file with the single item identified on the screen. It uses the Return Code/Data value in the structure to provide the single byte Return Reason code (field 6 of record 31). See section 14.6 of the DSTU X9.37-2003 specification for a comprehensive list of Return Reason codes.

The numeric value of the Sequence field in the input parameters is used as the Endorsing Bank Item Sequence Number in field 5 of the second record 35.

The Write Return Item function generates a _COPY version of the items specified by the item into the Working Directory. This file is then converted into a return file in the Working Directory with an extension of .R37.

Example:

Screen Parameters

Working Directory: Working directory file created prior to starting
c:\tempdemotest

File Name: The name of the file from which to retrieve the data.
c:\tempax9\temp.nsf.937

Cash Letter: Specifies which existing cash letter from which to retrieve the data.
1

Bundle: Specifies which existing bundle from which to retrieve the data.
1

Item: Specifies which existing forward item from which to retrieve the data.
2 (Contains a $9,000.00 item)

Record: Specifies which existing record from which to retrieve the data.
25

Return Code/Data
A

Creator Routing
987654320

Creator Date
20060622

Sequence (used as the Endorsing Bank Item Sequence Number in field 5 of the first Record 35. Assigned by the institution that creates the Return check.)
20

The new file will be located in the Working Directory and will have the same filename as the File Name name appended with the extension _COPY.937.R37. You should see one $9,000.00 item with a sequence of 20 in record type 35.

c:\tempdemotest\temp.nsf.937_COPY.937.R37 (View in X9Viewer)

Write Return Find List – Write Forward items from a list of Return Items

The Write Return function can also create a Forward Re-presentment X9.37 file from the Return items selected from the Pick List specified by Auxiliary File name. (Use triplet and Return Code in Pick List)

Each item is reformatted to that required for a re-presentment Forward 937 Item. The Pick List contains the single byte Return Reason code (field 6 of record 31), and is added into the Forward file (Record 28, Field 9). The file maintains the CL and Bundle structure information of the source file *but only contains the items in the PickList.*

The ECE values are filled from the parameters. They cannot be the value in the Return file. The numeric value of the Sequence field in the input screen parameters can be used as the Endorsing Bank Item Sequence Number in field 8 of the record 25.

Two record 28s are created. The first is created from the original record 35 and its return code. The second is a new endorsement record for this re-presentment.

The file will be rebalanced as a completely self contained Forward X9.37 file.
This file is then converted into a forward re-presentment file in the Working Directory with an extension of _Copy.937.R37.R37. If a file name is entered in the **Destination File parameter** the new Forward file will be placed there. It could be named with an extension of .937 if desired, ie., temp.nsf.937.

Example:

Screen Parameters

Working Directory: Working directory file created prior to starting
c:\tempdemotest

File Name: The name of the file from which to retrieve the data.
c:\tempax9\temp.nsf.937_Copy.937.R37

Destination File: Conditional - The name of the file where the output 937 will be placed.
c:\temp3\temp.nsf.937

Pick File: The name of the file containing the list of Return Items that Forward Re-presentments will be created
c:\tempdemotest\temp,nsf.937 Find.Lst
(this is the output file of the Find Pick 9.37 function we created containing the cash letter, bundle, item number of a $6,000.00 item – see page 59 for creation of this file.)

Creator Routing – Routing number of the creating bank
987654320

Creator Date – Date return created
20090622

The new Forward Re-presentment file will be located in the Working Directory and will have the same filename as the source appended with the extension _COPY.R37.

c:\tempdemotest\temp.nsf.937_COPY.R37_Copy.937.R37 (View in X9Viewer)

Or the user can specify the final destination in the Destination File as indicated above.

Write Return Item – Write Forward item from a single Return Item

The Write Return Item function can also create a Forward Re-presentment X9.37 file from a single Return item identified on the screen. It uses the Return Code/Data value in the structure to provide the single byte Return Reason code (Record 28, Field 9).

The ECE values are filled from the parameters. They cannot be the value in the Return file. The numeric value of the Sequence field in the input screen parameters can be used as the Endorsing Bank Item Sequence Number in field 8 of the record 25.

Two record 28s are created. The first is created from the original record 35 and its return code. The second is a new endorsement record for this re-presentment.

The file will be rebalanced as a completely self contained Forward re-presentment X9.37 file with an extension of _Copy.937.R37 in the Working Directory.

Example:

Screen Parameters

Working Directory: Working directory file created prior to starting
c:\tempdemotest

File Name: The name of the Return file from which to retrieve the data.
c:\tempax9\temp.nsf.937_COPY.R37

Destination File: Conditional - The name of the file where the output 937 will be placed.
c:\temp3\temp.nsf.937

Cash Letter: Specifies which existing cash letter from which to retrieve the data.
1

Bundle: Specifies which existing bundle from which to retrieve the data.
1

Item: Specifies which existing forward item from which to retrieve the data.
1

Record: Specifies which existing record from which to retrieve the data.
31

Return Code/Data
A

Creator Routing
987654320

Creator Date
20060622

Sequence (used as the Endorsing Bank Item Sequence Number in field 8 of the first Record 25.)
20

The new Forward Re-presentment file will be located in the Working Directory and will have the same filename as the File Name name appended with the extension _COPY.R37.

c:\tempdemotest\temp.nsf.937_COPY.R37_Copy.937.R37 (View in X9Viewer)

The user can specify the final destination and it will be moved.

Write Return

The Write Return function creates a X9.37 file with the items selected from the Find List specified by Auxiliary File name. ***This function is a combination of the Find Pick 9.37 function and the Write Return using a List.***

Each item is reformatted to that required for a Return Item. If the Find List contains the Return Reason code (field 6 of record 31), it is added into the Return file. The file maintains the CL and Bundle structure information of the source file but only contains the items in the Pick List.

The file will be rebalanced as a completely self contained X9.37 file. The numeric value of the Sequence field in the input parameters is used as the Endorsing Bank Item Sequence Number in field 5 of the second record 35.

The Write Return Find List function automatically generates a _COPY version of the items specified by the Find List into the Working Directory. This file is then converted into a return file in the Working Directory with an extension of R37.

Example:

Screen Parameters

Working Directory: Working file created prior to starting

c:\tempdemotest

File Name: The name of the file from which to retrieve the data.

c:\tempax9\temp.nsf.937

Auxiliary File: A text file containing the values of the records to be returned.

c:\tempdemotest\picklist.txt
(This is a text file containing a value we are looking for. We already created this text file when executing Find Pick 9.37, so edit it and add an "A" as the last value, i.e., 000600000, a – we are looking for all $6,000.00 items to return and will return with a return-code of "A".)

Cash Letter: Specifies the cash letter we are searching.

1 (this is the Cash Letter record)

Record: Specifies the record containing the field to match.

25 (this is the item record)

Field: Specifies the field containing the data to match.

7 (this is the amount field)

Creator Routing

987654320

Creator Date:

20060622

Sequence:

20

Output Files created in the Working Directory containing the returned item:

C:\tempdemotest\temp.nsf.937_copy.R37 (View in X9Veiwer)

Duplicate Functions

Test Add Dup – Update the History file

The Test Add Dup is the method to maintain a file history.

The input file, *File Name*, is compared with a history file of all the previous files in the history file AuxFileName, which contains the "fingerprint" data for all the previously tested files..

If the new file does not match any of the previous files, it is added to the History file. If there is a match, the result is non-zero and the name of the duplicate file is returned in *Data* field on the screen.

If the History file does not exist, it will be created.

Example:

Screen Parameters:

File Name: The name of the file to check.
c:\tempax9\temp.nsf.937

Auxiliary File: the name of the file containing the list of file history.
c:\tempdemotest\history

Data: The name of the duplicate file found in the history file.
Returned with = 143 (File duplicate)

Output History file:
c:\tempdemotest\history

History file contents: *(View in Notepad)*
History Rev 01
20060706:2237, 5DAA96DDD6825603805BF29AD3F078C197E35C60,

Test Dup

Test Dup function is concerned with testing files for duplicates. A simple history file is created containing the *File Name name* and hash code for the file. The hash code will detect the same content in a file even with different file names. This allows the user to rapidly detect retransmissions of the same file.

The input file, *File Name*, is compared with a history file of all the previously tested files stored in the history file AuxFile*Name.*

If the new file does not match any of the previous files, it is added to the History file. If there is a match, the result is non-zero and the name of the duplicate is returned in *Data*.

When Test Add Dup is called for the first time, the History Name file will be created.

Example:

Screen Parameters:

File Name: The name of the file to check.
c:\tempax9\temp.nsf.937

Auxiliary File: the name of the file containing the list of file history.
c:\tempdemotest\history

Data: The name of the duplicate file found in the history file.
Returned with = 143 (File duplicate)

Image and File QA Functions

These functions provide critical analysis of the input 937 file. This includes verifying quality of the MICR data and the quality of the 937 file and the TIFF contained within the 937 file.

Differences Between Print Ready and Exchange Ready:

Both Print Ready and Exchange Ready have bits for Compression. Since they are used in different context, they have different meanings. The following will cover some of those differences.

Compression:

Exchange – the universally accepted compression is Group 4 Fax Compression.
Printing – it is whatever the printer software will accept.

Resolution:

Exchange – the universally accepted resolution is 200 dpi although 240 is also common.

Printing – resolution can be 200, 200 or whatever. However, it must be set and it must make sense. Incorrect resolution will often cause a failure of dimension. In other words, the image size will be too big or too small based on the pixel width/height and resolution per width/height.

Image Missing:

Exchange – it is perfectly correct within 9.37 to exchange items without images.

Printing – a check items cannot be printed as an IRD without the front and back image.

See ***the AX9Lib Software Development Tool Kit API Reference Guide*** included with this installation for an in-depth description of these functions and their differences.

Test Print Ready

The Test Print Ready function will test the X9.37 file and determine if it is properly formed for ***IRD printing***. The return value will indicate the status of the file. A zero value means that the file is ready to print. A non-zero value means that the file is not ready to print.

The Test Print Ready function generates an output file containing a Pick File of items that fail the Print Ready test. In addition to the standard Cash Letter, Bundle, Item information there is a status field containing detailed description of the type of problem found.

The IRD print ready test includes the following:

Checks must have images even though non-check items may not.

Image file must be B/W TIFF. The AMP IRD Print application will support printing from a variety of TIFF compression types.

File format must conform to X9.37. The AMP IRD Print function does support Record 61 for printing Credits/deposit slips(See Appendix).

The IRD print ready function will **NOT** report as an error numerous conditions that would prohibit exchange(See Test Exchange Ready test). The following are not reported in the Print Ready function:

Control records that do not balance.

Images which do not conform.

Print Ready Conformance Problems – the following table identifies the errors that will display in the Pick File (errors).

Description	Bit Number	Numeric mask
Compression type wrong	0	1
Resolution--Missing or Incorrect	1	2
Dimension--Size Conformance	2	4
Image Missing	3	8
File Format Error	4	16
Decompression Error	5	32
TIFF Tag/937 Miss Match	6	64

The Test Print Ready function will load the screen with an annotated description of the contents of the source X9.37 file. This will recount the number of images out of compliance as well as other possible problems**. (See Appendix C for example of screen display.)**

Example:

Screen Parameters:

Working Directory: *The name of the working directory*

c:\tempdemotest

File Name: The name of the file to test.

c:\tempax9\temp.nsf.937

Pick File: the name of the file where the errors will be placed

c:\tempdemotest\Pick file errors

Report File: the name of a XML report file for records that fail to conform to Tough TIFF or bad file items

c:\tempdemotest\Report file

Sample of Pick File output:

1,1,1,8, ”1001 “ (this means CL, Bundle, Item 1 has an image missing)

1,1,4,265, ”1004 “(this means CL 1, Bundle 1, Item 4 is not Gp 4 Compression)

See Appendix C for complete examples of Pick File, Report File, and screen display.

Test Exchange Ready

The Test Exchange Ready function will test the X9.37 file and determine if it is properly formed for general X9.37 exchange. If you use the same file as used with Print Ready, it is possible to get an Error=132 meaning there is a missing image.

The return value will indicate the status of the file. A zero value means that the file is ready to exchange. A non-zero value means that the file is not ready to exchange.

The exchange ready tests include the following:

Checks must have images even though non-check items may not.

Image file must be B/W TIFF and conform to Appendix F.

File format must conform to X9.37 with the possible addition of Record 61. Record 61 is NOT acceptable in files with the FRB options selected. Control records must balance.

Exchange Ready Conformance Problems - the following table identifies the errors that will display in the Pick File (errors). (The first set from Print ready still apply. The 2nd set are exclusively Exchange ready problems.)

Description	Bit Number	Numeric mask
Compression type wrong	0	1
Resolution--Missing or Incorrect	**1**	**2**
Dimension--Size Conformance	2	4
Image Missing	3	8
File Format Error	4	16
Decompression Error	5	32
TIFF Tag/937 Miss Match	6	64

Description	Bit Number	Numeric mask
Compression--Not Group 4	8	256
Strips--Not one	9	512
Resolution -- Not 200 dpi	10	1024
Fill Order --Not one	11	2048
Endian -- Not Little	12	4096
Sub File -- Not zero	13	8192
Not B/W	14	16384
Tag Missing	15	32768
Orientation -- Not one	16	65536
Compressed Size--Too big/small	17	131072

The Test Exchange Ready function will load the screen with an annotated description of the contents of the source X9.37 file. This will recount the number of images out of compliance as well as other possible problems**.**

Example:

Screen Parameters:

Working Directory: *The name of the working directory*

c:\tempdemotest

File Name: The name of the file to check.

c:\tempax9\temp.nsf.937

Pick File: the name of the file where the errors will be placed

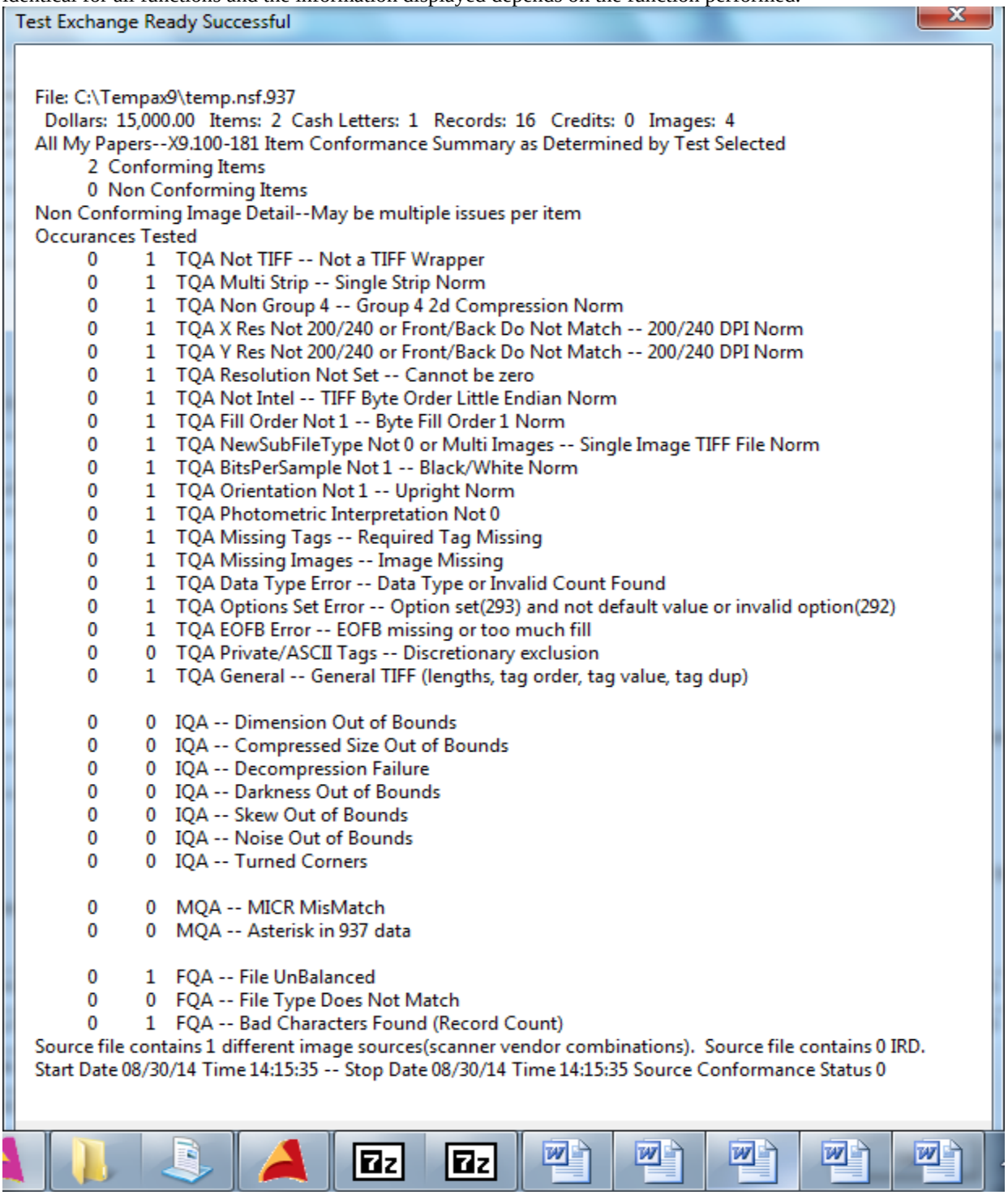
c:\tempdemotest\Pick file errors

Report File: the name of a XML report file with items that fail to conform to Tough TIFF or bad file items

c:\tempdemotest\Report file

Appendix contains more detailed errors messages. Below is example report.

Screen Display – Test Exchange Ready Error Report - The format of this report is identical for all functions and the information displayed depends on the function performed.



Not Correctable List

The Not Correctable List function creates a subset list with 937 items that are not correctable.

It will read a pick list file, determine which items are correctable to the X180 TIFF standard, and then write out a new pick file which lists just the not correctable items.

The output list can then be used along with the List Delete function to edit out those items from an X9.37 file prior to using Convert From 937 to correct the non-conforming check images.

File Name: The name of the input pick list file that will be read. This is initially created from the ConvertFrom937 operations

c:\testfiles – x9qualifier\2K testfile.937

Working Directory: *The name of the working directory*

c:\tempdemotest

Pick File: the name of the file where the errors will be placed..

c:\tempax9\image non correctable.lst

Sample of Pick File output:

1,1,1,8, ”1001 “ (this means CL, Bundle, Item 1 has an image missing)

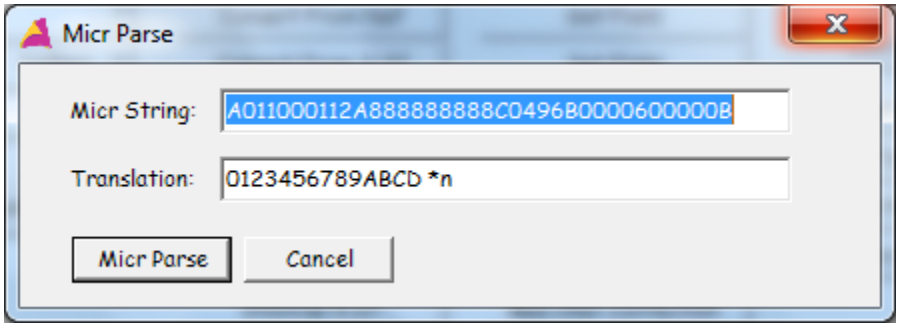
1,1,4,265, ”1004 “(this means CL 1, Bundle 1, Item 4 is not Gp 4 Compression)

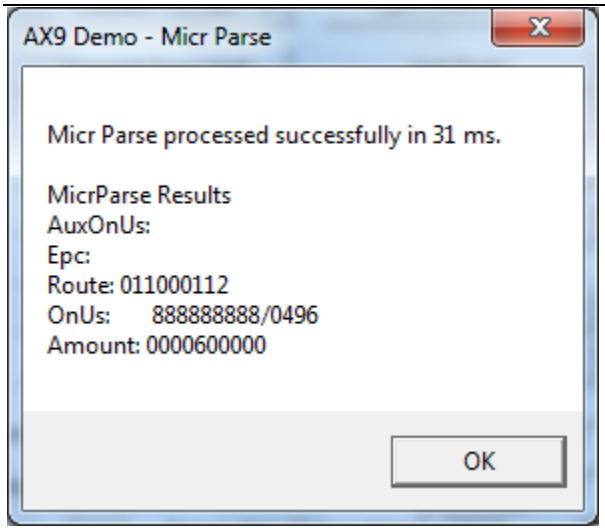
Miscellaneous Functions

This set of functions provides general utility processes. MicrParse only uses the source and translator strings to determine the micr fields, and does not read an image. MicrParse, however, can be called after reading the MICR line from an image.

MICR Parse

This function parses a MICR line from a string and displays on the screen.



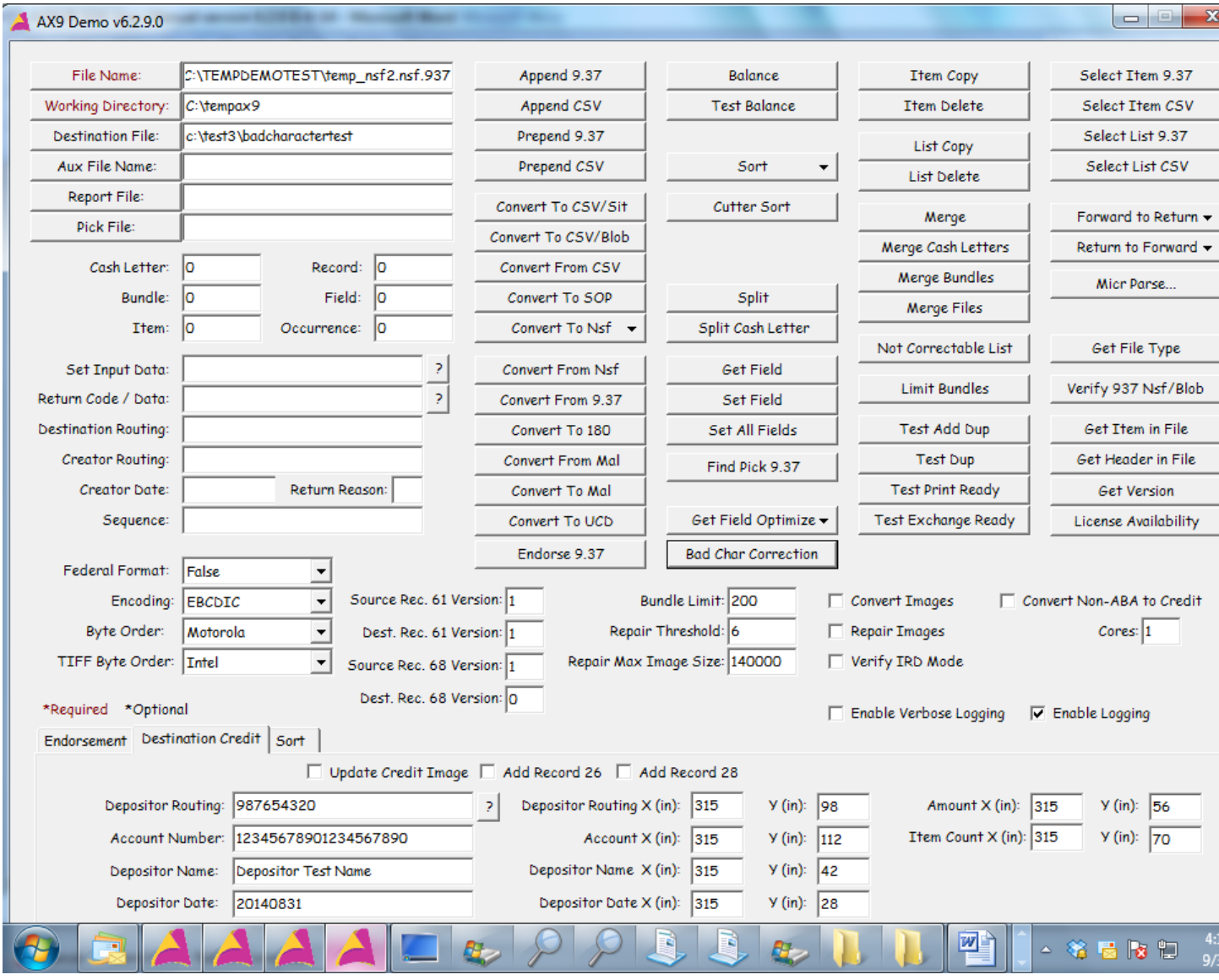


Bad Char Correction

Tests for bad characters and creates a report.

If a Destination File is specified, corrects bad characters and generates a new file with the corrected characters. Replaces the bad char with an '*'. Should be able to view this file in X9Viewer.

The Server bit in the AX937LIB license is required.



Get File Type

The Get File Type function allows the user to determine the type of input file in order that the transmission of return files be in the same form as the input without detailed processing of the X9.37 file.

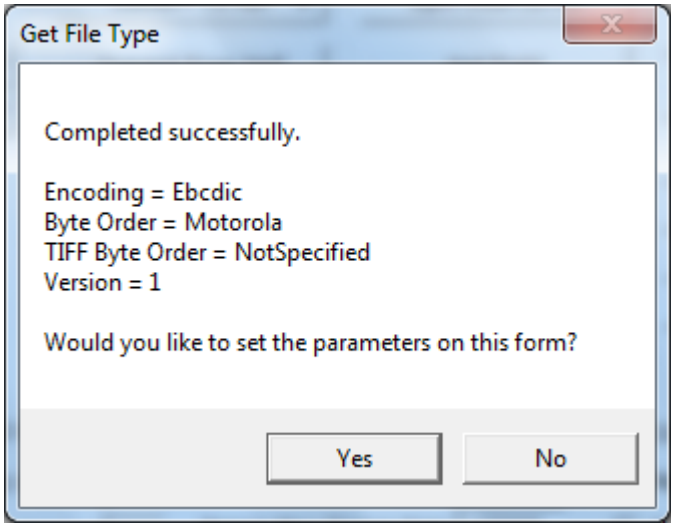
When using the CSV format to load and unload a X9.37 file, this function call provides the necessary information to rebuild a new X9.37 with the same data types as was in an original source file.

Example:

Screen Parameters:

File Name: c:\tempax9\temp.nsf.937

Screen displays:



Get Items in File

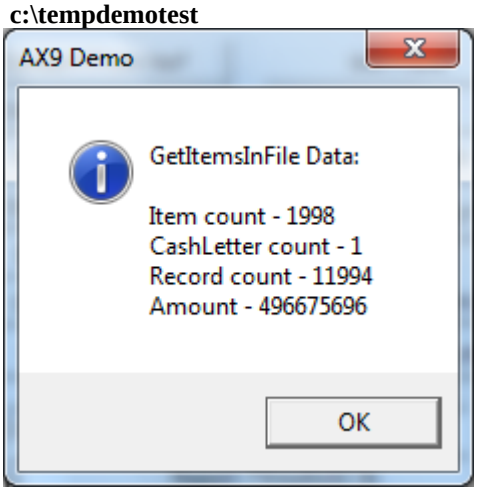
The Get Items in File function allows the user to determine the numeric contents of record

99

in the specified x9.37 file.

File Name: The name of the input file
c:\tempax9\test.937

Working Directory:
c:\tempdemotest



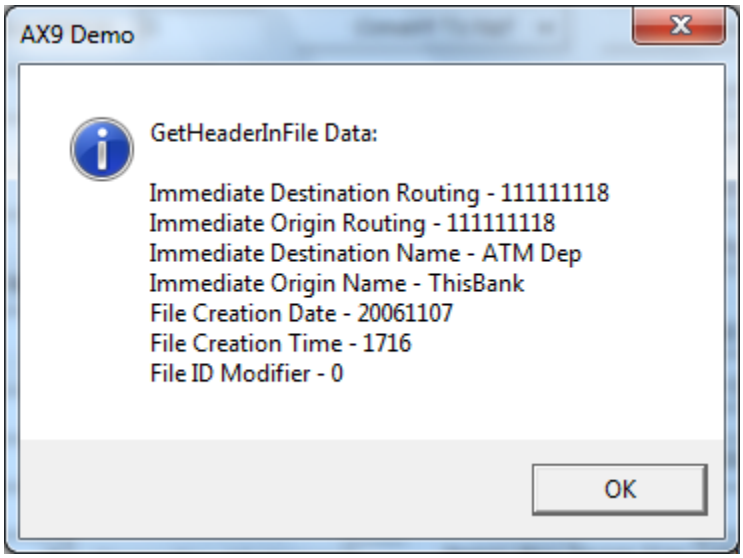
Get Header in File

This function is used to return the string contents of Record 01 in the specified X9.37 file.

Example:

Screen Parameters:

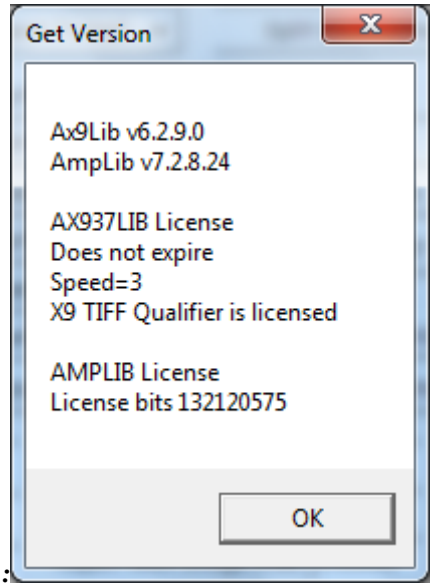
File Name:
c:\tempax9\temp.nsf.937
Working Directory:
c:\tempdemotest



Get Version – No input parameters, just click button

The Get Version function returns to the user the version level and sub levels of the underlying DLL. If you need to contact the AMP Software Developers, they will require this information.

When you click OK, It will also display several more screens giving the license expiration date, whether a license for X9 Tiff Qualifier is present and the license bits of the AmpLib.



:

Verify 937 Nsf/Blob – read a X9.37 file and verify MICR values

Verify 937 Nsf/Blob function will read an X9.37 file and verify the MICR values of the front check images with the MICR values contained in the X9.37 file and output a NSF/BLOB file.

This method will read an X9.37 file entered in File Name name and generate a number of NSF/BLOB files in the working directory.

The check images pointed to by the NSF/BLOB files will then be recognized using MICRBatch/AmpLib and the resultant MICR codes will be compared against the machine read values from the X9.37 file.

Those checks that match the source X9.37 file will be included in an output NSF/BLOB file that will be placed in the working directory. This output file will have the extension MOC and the file that has the check images that did *not* verify will have the extension CMP.

Example:

Screen Parameters

Working Directory: Working file

c:\tempdemotest

File Name: Input file

c:\tempax9\temp.nsf.937

Verify IRD Mode:

Check mark

Output - Files created in the Working Directory:

C:\tempdemotest\temp.nsf.937.NSF **(View in notepad)**

C:\tempdemotest\temp.nsf.937.MOC **(View in notepad)**

(Checks verified correctly)

C:\tempdemotest\temp.nsf.937.CMP **(View in notepad)**

Endorse 937

Endorse 937 function will read an X9.37 file and annotate every reverse-side check image in the X9.37 file with a custom endorsement.

The Endorse 937 function will draw an endorsement consisting of a box containing 4 lines of text either in portrait (normal left-right top-down) or landscape (rotated 90 degrees clockwise) format. The location and format of the endorsement are set using the Endorsement Options. **Note:** You can use the default parameters by checking the boxes on the right side of the screen instead of setting them individually.

Example:

Screen Parameters

Working Directory: Working file - must already exist
c:\tempdemotest

File Name: Input file - the 937 file to which endorsements will be created
c:\tempax9\temp.nsf.937

Enter Endorsement Options:
Bank Name: **ABC Bank**
Bank Routing: **123456780**
Deposit Account Number: **12345678987654321**
Deposit Name: **John Doe**

Font Style: **Arial**
Drawing Options: **w=400s=12j=c**
Each of these are redundant, as they are equivalent to the default values

Endorsement Height:1
Endorsement Width: **1.5**

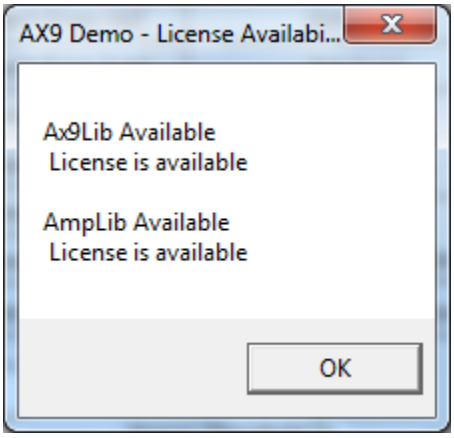
Bottom Offset: **1**
Right Offset**:** **.75**

Endorse 937
Deposit/Payee endorse - (check mark = rotates to left right top-bottom)

Output - Files created in the Working Directory:
C:\tempdemotest\temp.nsf.937 **(View in X9Viewer)**

Get License Availability

This function indicates whether the AX9LIB and AMPLIB licensing is available.



APPENDEX A - INPUT FILE FORMATS

An interface process must be created from your check scanning process to the AMP X9 products. This would include reading the check image files and creating one of the following formats for input into X9 processing.

After the input file is created, enter AX9Lib and use the correct "Convert From" function to create the appropriate X9.37 file. If you are using a X9.37 as input, you would use the correct Convert To function to create the various output files available.

If your output file is a X9.37 or CSV, you can display in AMP's X9Viewer to analyze, update or correct errors.

AX9 processing will accept the following input formats:

1. NSF (Normalized Scanner Format) – There is four NSF formats for creating X9.37 files. The first line of each has the file type information and additional other information that may be needed by the file. This could include a file name and/or translation table.

   The Convert From NSF function will automatically detect the various NSF formats.

   - NSF – AMP's AX9lib function Convert From NSF will read a NSF formatted file and construct a X9.37 file in the working directory. It can be created by using editing software such as Notepad.

     For this function to finish successfully, all of the Single Image TIFF or JPEG files referenced in the source NSF file must exist and be valid TIFF or JPEG files. Each image is represented by a single TIFF file with one image in the file. The name of the file is stored in the NSF file. The NSF format has a comma separated item for each X9.37 MICR line field.

     The ECE Institution Item Sequence Number may be included on the NSF line after the other data. If the NSF data does not have Item Sequence Number data, then the numeric value in the parameter Sequence will be incremented and used as the ECE Institution Item Sequence Number for entry in field 8 of the first record type 25. The NSF file generated by the X9.37 to NSF also contains the Cash Letter, Bundle and Item numbers, but that information does not need to be present for input.

   - NSF2 – The NSF2 is the format that would typically come from a scanner. This format only contains the front and back image file names and the MICR line data. The full MICR line is input as a single field and the software parses into the proper fields. This format requires that the first line of the file contain a client's translation table so that the special MICR codes can be represented in text. There is no AX9LIB function to generate an output NSF2 format.

   - NSF/MIT - Multi Image TIFF format for NSF files. The source of the front and back side image data is a multi image Tiff file.

When the check images are stored in a Multi Image Tiff file, the NSF format must be able to identify that it is a MIT file. Therefore, the header line in this case is NSFMIT.

For each check to be included in the file, a single text line is needed in the NSF/MIT file. The source line contains the file names for TIFF images of the check front, its index and the file name for the back image and its index. It also contains the MICR line data in a simple parsed format. There are a few parameters added by the calling function to identify source and destination routing numbers and a few other elements. However what comes from the scanner is the critical data. The parsed MICR line follows the X9.37 and X9.90 definitions of MICR line parsing. *NOTE: This function requires a special license.*

- NSF/BLOB – A "BLOB" is a generic usage that would be an unformatted concatenation of data. This format does not have individual file names for each image, but has only a reference to the location and length of the image in a master BLOB whose name is in the NSF header line of the file and is NOT repeated with each line that uses it. Each image is represented by a reference offset and size into a multi image BLOB. *NOTE: This function requires a special license.*

2. CSV Format – Comma Separated Variable – this is a comma delimited file with a one to one correspondence with X9.37 format as specified in the DSTU X9.37 manual. Many times this is used to output a full X9.37 file to a CSV for editing in a text editor or via software and then the edited CSV file is used to rebuild a new X937 file. An example is included with the installation package.

3. X9.37 Format – This format is necessary for electronic exchange as specified in the DSTU X9.37-2003 Manual for specifications for the X9.37 file. A X9.37 formatted file can also be used as input to create various classes of output files for further dissection of contents. See Output section for file formats that can be created from the X9.37 format, including NSF/SIT, NSF/BLOB, CSV/SIT, CSV/BLOB, SOP, 180, and MAL.

4. MAL – This is a FRB Payers Services formatted ASCII file. The title is MAL because the header records are "mal formed" 937 formatted files from a version of a 1994 - X9.37. It is the same as the current X9.37 format but without the image records and a wrong value for the record length field. *NOTE: This function requires the X9180 license feature bit.*

NSF FILE FORMAT

NSF is a format of All My Papers. The NSF format is probably the simplest format to use and the easiest to create for product verification. There is only one line item per check image using this format and a single image TIFF image per line.

The following shows an example of a NSF file that could be created from a scanned image prior to entering into AMP's AX9Lib and an example inserting a Credit Record 61.

Example 1 - NSF Input Text File

```
NSF," ",0,0,"Copyright 2005 All My Papers"
C:\temp\00000008.tif","C:\temp\00000010.tif"," "," ","011000112","       77777777/0377","0000600000","011000112
C:\temp\00000015.tif","C:\temp\\0000017.tif"," "," ","011000112","       88888888/0596","0000900000"
```

Field Name	Data Length	Field Value Examples
File Header Record		
File Type	Alphanumeric Variable	NSF,
Input File Name	Alphanumeric Variable	" ",
Reserved	Numeric	0,
Reserved	Numeric	0,
Proprietary Information	Alphanumeric Variable	"Copyright 2005 All My Papers"
Item Record:		
Front Side Image File Name	Alphanumeric Variable	"C:\temp\0000008.tif",
Back Side Image File Name	Alphanumeric Variable	"C:\temp\0000010.tif",
MICR Aux On US	Alphanumeric, Blank (right-justify)	" ",
MICR EPC	Alphanumeric, Blank	" ",
MICR Routing Number	Alphanumeric	"011000112",
MICR On Us	Alphanumeric, Blank (right-justify)	" 77777777/0377",
MICR Item Amount	Numeric (Zero-fill)	"0000600000",
ECE Institution Item Seq Nbr	NB	"01100012 ", (if any)
Payee Name	Alphanumeric	"John Doe ",
Deposit Account Number	NB	"123456789",
Branch	NB	" 8888",

Example 2 - NSF Credit Record

```
NSF," ",0,0,"Copyright 2006 All My Papers"
"REC+61+VER+01","c:\temp\00000038.tif","c:\temp\00000041.tif","000000000000151","1","123456780","0000000000561016718
0","0000900000","00000000001500K"
C:\temp\00000008.tif","C:\temp\00000010.tif"," "," ","011000112","       77777777/0377","0000900000","011000112
```

Field Name	Data Length	Field Value
Credit Item Record:		
Record 61 type	Mandatory	"REC+61+VER+01",
Front Side Image File Name	Alphanumeric Variable	"C:\temp\0000038.tif",
Back Side Image File Name	Alphanumeric Variable	"C:\temp\0000041.tif",
MICR Aux On US	Alphanumeric, Blank	"000000000000151",
MICR EPC	Alphanumeric or Blank	"1",
MICR Routing Number	Alphanumeric	"123456780",
MICR On Us	Alphanumeric, Blank	"00000000005610167180",
MICR Item Amount	Numeric	"0000900000",
BOCL Item Sequence	Numeric	"00000000001500K"

Example 3 - NSF Input Text File with a "28" parameter

```
NSF,"H:\All My Papers\X9.37\Copy of Fed\ForwardSample[1].txt",0,0,"Copyright 2005 All My
Papers","28"
C:\temp\00000008.tif","C:\temp\00000010.tif"," "," ","011000112","       77777777/0377","0000600000","011000112
C:\temp\00000015.tif","C:\temp\\0000017.tif"," "," ","011000112","       88888888/0596","0000900000"
```

Field Name	Data Length	Field Value Examples
File Header Record		
File Type	Alphanumeric Variable	NSF,
Input File Name	Alphanumeric Variable	" ",
Reserved	Numeric	0,
Reserved	Numeric	0,
Proprietary Information	Alphanumeric Variable	"Copyright 2005 All My Papers"
Parameter	Numeric	"28"
Item Record:		
Front Side Image File Name	Alphanumeric Variable	"C:\temp\0000008.tif",
Back Side Image File Name	Alphanumeric Variable	"C:\temp\0000010.tif",
MICR Aux On US	Alphanumeric, Blank (right-justify)	" ",
MICR EPC	Alphanumeric, Blank	" ",
MICR Routing Number	Alphanumeric	"011000112",
MICR On Us	Alphanumeric, Blank (right-justify)	" 77777777/0377",
MICR Item Amount	Numeric (Zero-fill)	"0000600000",
ECE Institution Item Seq Nbr	NB	"01100012 ", (if any)

NSF2 FILE FORMAT

NSF2 is a format of All My Papers. The NSF2 format provides more user flexibility. There is only one line item per check image using this format. This format only contains the file names for the TIFF front and back image file names and the MICR line data. This format requires that the first line of the file contain a translation table so that the special MICR codes can be represented in text. See AX9Lib/Help/AX9Lib Reference Manual for further definition of Translation Table.

NSF2 Translation Table

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,15,16,17

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ⑆, ⑇, ⑈, ⑉, , ? ,NA

(Route Code, Amount Code, Onus Code, Dash Code, Blank, Unknown, NA)

Example - NSF2 Input File

The user must supply a Translation Table in the File Header Record to show how the MICR special characters relate to the input.

```
NSF2," ","0123456789ABCD *n","Copyright 2005 All My Papers"
"c:\temp\00000008.tif","c:\temp\00000010.tif",A011000112A777777777C 0377B0000900000B,"111111111111111"
"c:\temp\00000015.tif","c:\temp\00000017.tif",A011000112AC88888888C 0496B0000600000B,"111111111111111"
```

Field Name	Data Length	Field Value Examples
File Header Record		
File Type	Alphanumeric Variable	NSF2,
Input File Name	Alphanumeric Variable	" ",
Translation Table	Alphanumeric	"0123456789ABCD *n",
Proprietary Information	Alphanumeric Variable	"Copyright 2005 All My Papers"
Item Record:		
Front Side Image File Name	Alphanumeric Variable	"C:\temp\0000008.tif",
Back Side Image File Name	Alphanumeric Variable	"C:\temp\0000010.tif",
MICR Line		A011000112A777777777C 0377B0000900000B,
ECE Item Sequence Number	Alphanumeric(15)	"111111111111111"

NSFMIT FILE FORMAT

The NSFMIT format is a multi image file and supports all of the normal NSF fields plus it allows for input of the Payee and Deposit Account number as seen on the last line of this example. When the check images are stored in a Multi Image Tiff file, NSF format must be able to identify that it is a Multi Image file and the image index within the multi image Tiff file. The header line in this case is NSFMIT.

Example - NSFMIT

NSFMIT,"",0,0,"Copyright 2005 All My Papers"

"C:\Image file.tif",1,,,"C:\\Image file.tif",2,,," "," ",""," 77777777/0377","0000900000","11111111111"

"C:\Image file.tif",3,,,"C:\Image file.tif",4,,," "," ","011000112","88888888/0596","0000600000","2222222222222"

"C:\Image file.tif",7,,,"C:\Image file.tif",8,,," "," ","661866434","6718669047/0006641","0000000400",

"555555555555555","Franklin","232323"

Field Name	Data Length	Field Value Examples
File Header Record		
File Type	Alphanumeric Variable	NSFMIT,
Input File Name	Alphanumeric Variable	“”,
Reserved	Alphanumeric	0,
Reserved	Alphanumeric	0,
Proprietary Information	Alphanumeric Variable	“Copyright 2005 All My Papers”
Item Record:		
Front Side Image	Alphanumeric Variable	“C:\Image file.tif”,
Front Side Index	Numeric	1,
Reserved		,
Reserved		,
Back Side Image	Alphanumeric Variable	“C:\Image file.tif”,
Back Side Index	Numeric	2,
Reserved		,
Reserved		,
MICR AUX On Us		“ “,
MICR EPC		“ ”,
MICR Routing		“”,
MICR ONUS		“ 7777777/0377”,
MICR Amount		“00009000000”,
ECE Institution Item Seq Nbr		“11111111111”,
Payee		
Deposit Account Number		

NSFBLOB FILE FORMAT

A “BLOB” is a generic usage that identifies an unformatted concatenation of data. This format does not have individual file names for each image, but has only a reference to the location and length of the image in a master BLOB whose name is in the NSF header line of the file and is not repeated with each line that uses it. Each image is represented by a reference offset and size into a multi image BLOB.

```
NSFBLOB,"",0,0,"Copyright 2005 All My Papers"
0000000625,0000007036,"36DA29E25F78EF5392F39A5ABCD8704EA908D651",0000007950,0000003330,"0622D
9A681267A9D8225E80120A2C1DC058C669C","               "," ","011000112","
77777777C0377","0000900000","987654320"

0000011737,0000005620,"6C4996B783E58B2F1E08AF3CC2DA7AD1C3CDB635",0000017646,0000003082,"E28EF
F95051228F4943CD0A66D1702441852DFF5","               "," ","011000112","
88888888C0596","0000600000","987654320","20090721","011000112
```

Field Name	Data Length	Field Value Examples
File Header Record		
File Type	Alphanumeric Variable	NSFBLOB,
Input File Name	Alphanumeric Variable	“ ”,
Reserved	Alphanumeric	0,
Reserved	Alphanumeric	0,
Proprietary Information	Alphanumeric Variable	“Copyright 2005 All My Papers”
Item Record:		
Front Side Image Index	Numeric	0000000625,
Front Side Image Length	Numeric	0000007036,
Front Side Image	Alphanumeric	“36DA29E25F78EF5392F39A5ABCD8704EA908D651”,
Back Side Image Index	Numeric	0000007950,
Back Side Image Length	Numeric	0000003330,
Back Side Image	Alphanumeric	“0622D9A681267A9D8225E80120A2C1DC058C669C”,
MICR AUX On Us		“ “,
MICR EPC		“ ”,
MICR Routing		“01000112”,
MICR ONUS		“ 7777777/0377”,
MICR Amount		“000090000000”,
ECE Institution Item Seq Nbr		“987654320”,

CSV FILE FORMAT

In CSV format there is a **one to one** correspondence between each field in the X9.37 file and a comma separated field in the CSV file. The exceptions are the variable length fields such as image data. The CSV file will have an image reference or the name of a file where the image data is placed.

Definitions are only given for a few of the records necessary. AMP will supply Records 10, 20, 26, 28, 70, 90, and 99. If records 10 and 20 are not supplied, the file will process, but will not have the necessary valid fields for exchange and will need to be manually edited.

A Credit Record 61 can also be inserted in this file format.

The CSV file can be edited via Notepad. **Note**: Excel does not convert well for use with AX9Lib. The example shows a file similar to what is included with the AX9Lib installation.

Example - CSV Input Text File

```
"'01","'03","'T","'011000015","'011000112","'20040916","'0909","'N","'John Doe Bank     ","'Presidents Bank   ","'1","'  ","
"'10","'01","'011000015","'011000112","'20040809","'20040812","'0431","'I","'G","'10111011","'John Smith    ","'1234567890",
"'20","'01","'011000015","'011000112","'20040809","'20040809","'BU10111011","'0001","'3 ","'011000112","'     ","'
"'25","'              ","' ","'01100011","'2","'       77777777/0377","'0000900000","'210041944      ","'G","'1","'1","'Y",
"'26","'1","'011000015","'20040809","'210041940      ","'                  ","'     ","'               ","'Y","'2","'1","' "
"'28","'01","'011000112","'20040809","'210041940      ","'Y","'2","'1","' ","'              ","'                "
"'50","'1","'301171285","'20040809","'00","'00","'0007036","'0","'00","'0","'00","'00000","'0000000","'0000000","'0","'
"
"'52","'211870870","'20040809","' 3","'210041944      ","'                  ","'                  ","'
","'0","'0000","'0000","'0000","'0000","'   0","'","'    0","'","'   7036","'C:\temp\00000008.tif"
"'50","'1","'301171285","'20040809","'00","'00","'0003330","'1","'00","'0","'00","'00000","'0000000","'0000000","'0","'
"
"'52","'211870870","'20040809","' 3","'      210041944","'                  ","'                  ","'
","'0","'0000","'0000","'0000","'0000","'   0","'","'    0","'","'   3330","'C:\temp\00000010.tif"
"'70","'0002","'000001500000","'           ","'00004","'                  ","'                  "
"'90","'000001","'00000002","'00000000900000","'000000004","'Presidents Bank   ","'20040809","'               "
"'99","'000001","'00000020","'00000002","'0000000001500000","'John Smith    ","'1234567890","'               "
```

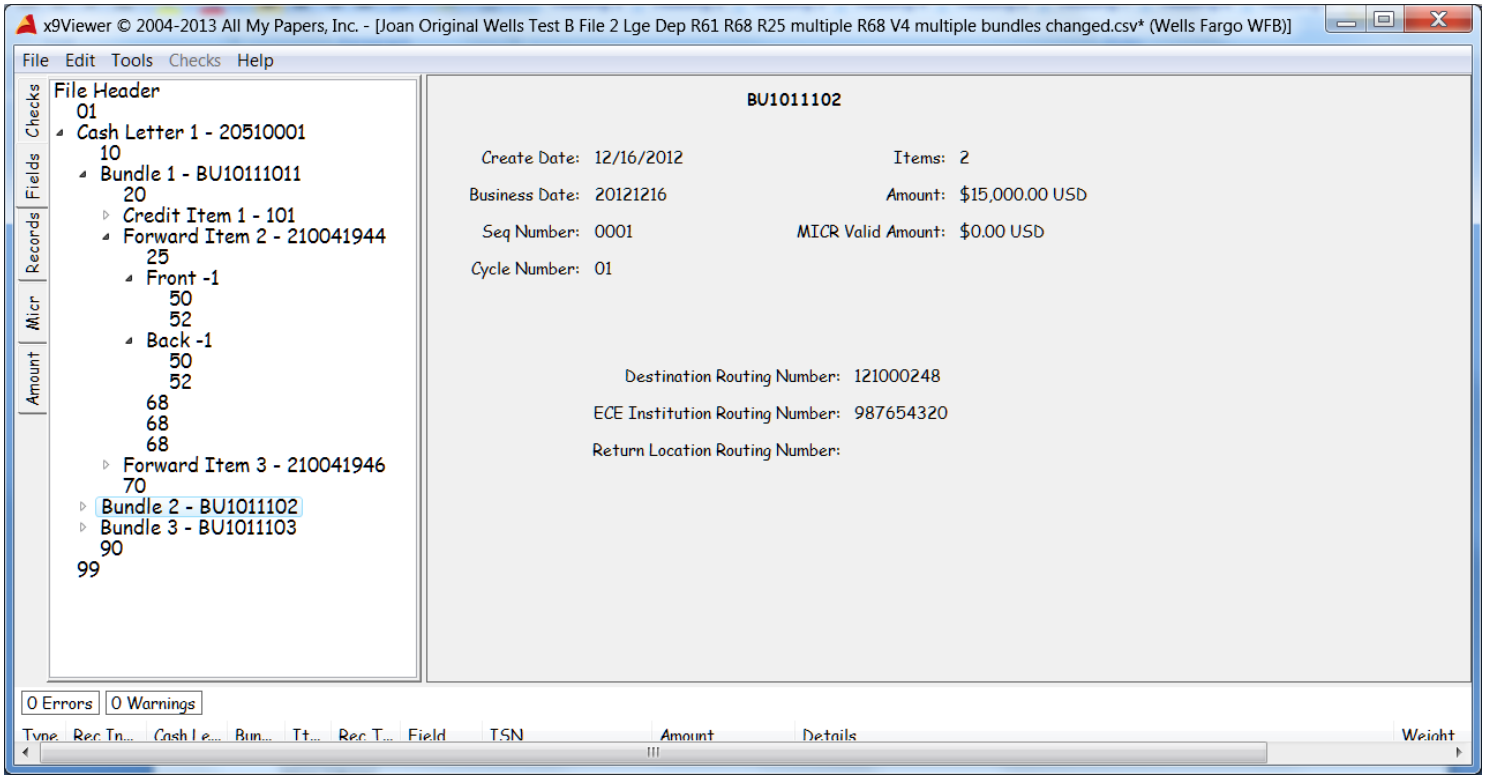
Field Name	Data Length	Field Value
File Header Record		
Record Type	Numeric	"'01',
Standard Level	Numeric	"'03",
Test File Indicator	Alphabetic	"'T",
Immediate Destination Routing Number	Numeric	"'011000015",
Immediate Origin Routing Number	Numeric	"'011000112",
File Creation Date	Numeric	"'2004080916",
File Creation Time	Numeric	"'0909",
Resend Indicator	Alphabetic	"'N",
Immediate Destination Name	Alphabetic	"'John Doe Bank",
Immediate Origin Name	Alphabetic	"'Presidents Bank",
File ID Modifier	Alphanumeric	"'1",
Country Code	Alphanumeric	"' ",
User Field	Alphanumeric	"' ",
Reserved	Alphanumeric	"' "

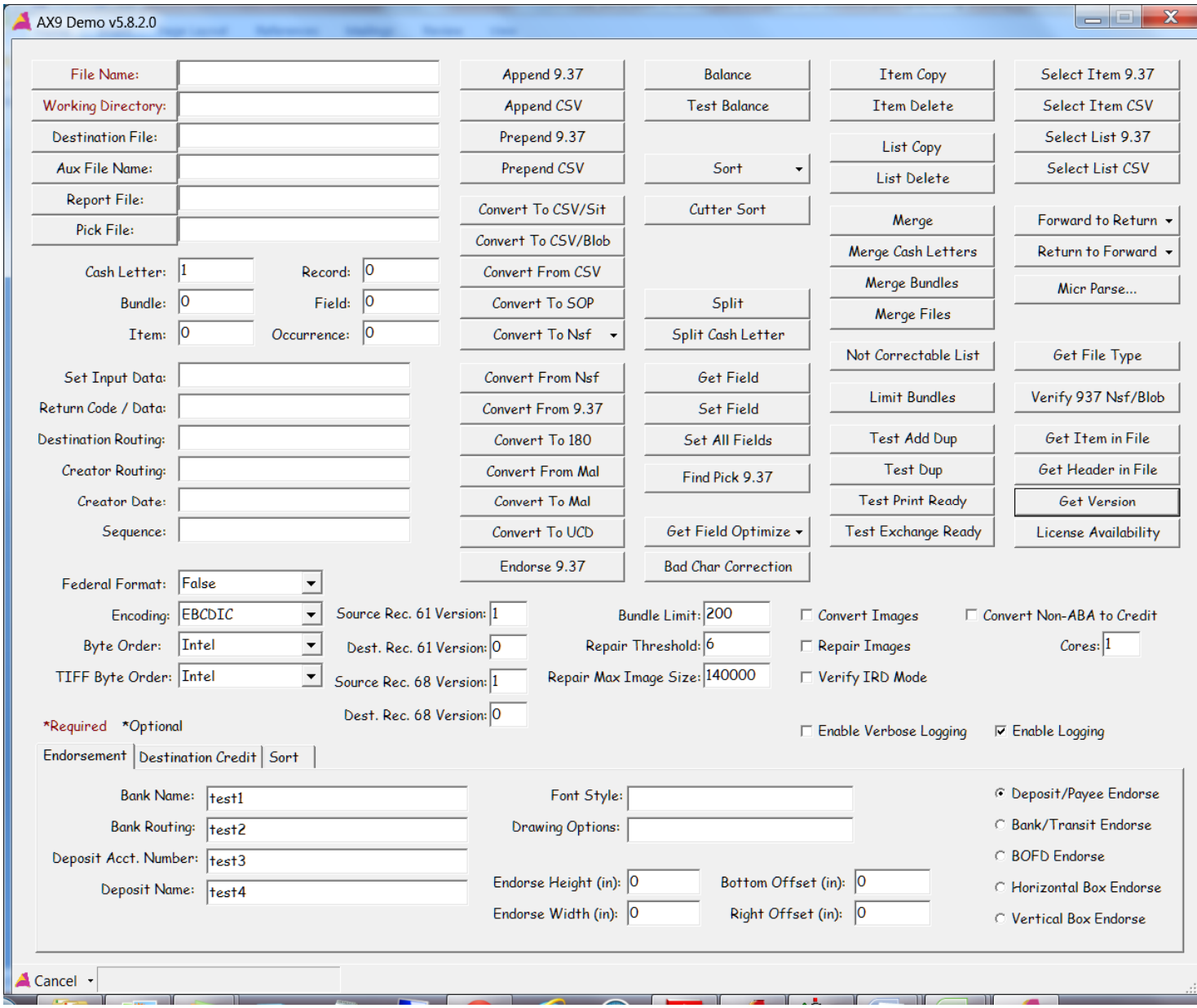
Field Name	Data Length	Field Value Examples
Record 25 – Image View Detail		
Record Type	Numeric	"'25',
Micr Auxiliary On-Us	NUBSM	"' ",
Micr EPC	ANS	"' ",
Micr Payor Bank Routing Number	Numeric	"'01100011",
Micr Payor Bank Routing Number	Numeric	"'2",
Micr On-Us	NBSM	"' 77777777/0377",
Micr Item Amount	N	"'0000900000",
ECE Institution Item Sequence Nbr	NB	"'210041944 ",
Documentation Type Indicator	AN	"'G",
Return Acceptance Indicator	AN	"'1",
MICR Valid Indicator	Numeric	"'1",

BOFD Indicator	Alphabetic	"'Y",
Check Detail Record Addendum Count	Numeric	"'02" ,
Correction Indicator	Numeric	"'4" ,
Archive Type Indicator	Alpha	"'F"

X9.37 FILE FORMAT

This format is necessary for electronic exchange as specified in the DSTU X9.37-2003. It can also be used as an input format into AX9Lib to create various output files for additional processing. Output formats created from the X9.37 format, including NSF/SIT, NSF/BLOB, CSV/SIT, CSV/BLOB, SOP, 180, and MAL.




AX9 Demo v5.8.2.0
File Name:
Working Directory:
Destination File:
Aux File Name:
Report File:
Pick File:
Cash Letter: 1
Record: 0
Bundle: 0
Field: 0
Item: 0
Occurrence: 0
Set Input Data:
Return Code / Data:
Destination Routing:
Creator Routing:
Creator Date:
Sequence:
Federal Format: False
Encoding: EBCDIC
Byte Order: Intel
TIFF Byte Order: Intel
*Required *Optional
Append 9.37
Append CSV
Prepend 9.37
Prepend CSV
Convert To CSV/Sit
Convert To CSV/Blob
Convert From CSV
Convert To SOP
Convert To Nsf
Convert From Nsf
Convert From 9.37
Convert To 180
Convert From Mal
Convert To Mal
Convert To UCD
Endorse 9.37
Balance
Test Balance
Sort
Cutter Sort
Split
Split Cash Letter
Get Field
Set Field
Set All Fields
Find Pick 9.37
Get Field Optimize
Bad Char Correction
Item Copy
Item Delete
List Copy
List Delete
Merge
Merge Cash Letters
Merge Bundles
Merge Files
Not Correctable List
Limit Bundles
Test Add Dup
Test Dup
Test Print Ready
Test Exchange Ready
Select Item 9.37
Select Item CSV
Select List 9.37
Select List CSV
Forward to Return
Return to Forward
Micr Parse...
Get File Type
Verify 937 Nsf/Blob
Get Item in File
Get Header in File
Get Version
License Availability
Source Rec. 61 Version: 1
Dest. Rec. 61 Version: 0
Source Rec. 68 Version: 1
Dest. Rec. 68 Version: 0
Bundle Limit: 200
Repair Threshold: 6
Repair Max Image Size: 140000
Convert Images
Repair Images
Verify IRD Mode
Convert Non-ABA to Credit
Cores: 1
Enable Verbose Logging
Enable Logging
Endorsement
Destination Credit
Sort
Bank Name: test1
Bank Routing: test2
Deposit Acct. Number: test3
Deposit Name: test4
Font Style:
Drawing Options:
Endorse Height (in): 0
Endorse Width (in): 0
Bottom Offset (in): 0
Right Offset (in): 0
Deposit/Payee Endorse
Bank/Transit Endorse
BOFD Endorse
Horizontal Box Endorse
Vertical Box Endorse
Cancel

MAL FILE FORMAT

The MAL file format is a Federal Reserve Bank (FRB) Payers Services formatted ASCII file. The title is MAL because the header records are “mal formed” 937 format. Using the function Convert From MAL will read this format and generate an exchange ready 937 file.

Example - MAL Input Text File

```
&0103T12345678098765432020090721l924N                                    1
&1001123456780987654320200907212009072119 24IGADVT2436                      C
&2001123456780987654320200907212009072100000310270001000000000000
&25                 011000112       77777777/0377000090000011000112      G01Y01
&261987654320200907210110001l2      210041944              20040809       Y
&25                 011000112       88888888/0596000060000011000112      G01Y01
&261987654320200907210110001l2      210041946              20040809       Y
&70000200000150000000000150000000000
&900000010000000200000001500000000000000
&990000010000001000000002000000001500000
```

APPENDEX B - Output FILE FORMATS

The following output files can be created by using a X9.37 formatted file as the source and using the correct function to convert.

1. X9.37 Format - Format necessary for electronic exchange as specified in the DSTU X9.37-2003 Manual for specifications for the X9.37 file. An output X9.37 can be created from any of the Convert From functions excluding Convert From Mal.

2. CSV/SIT – Comma Separated Variable – A X9.37 file is read to build a CSV file with a one to one correspondence with X9.37 format as specified in the DSTU X9.37 manual. Primarily used to output a full X9.37 file for editing in a text editor or via software and then the edited file is used to rebuild a new X9.37 file. Each record in the X9.37 file generates a single text line in the CSV file. The captured images are stored as TIFF files and the text data placed in the CSV format.

3. CSV/BLOB – This function will read a X9.37 file and generate an output file in the working directory. The image files are pointed to in the CSV Blob format and not moved from the original source X9.37 file. The CSV file will have the same name as the original file appended with a .CSV extension.

4. SOP - The Convert to SOP function will read an exchange ready X9.37 file and generate an output SOP IV.8 formatted file in the working directory. *Warning: The SOP IV.8 file format is described with limited detail in a FRB document. NOTE: This function requires the X9180 license feature bit.*

5. NSF/SIT – The function Convert To NSF/SIT function will read a X9.37 file and construct a NSF file in the working directory.

6. NSF/BLOB – A X9.37 file can be read to create a NSF/BLOB output format. This is a Multi Image format for NSF Files. A "blob" is a generic usage that is an unformatted concatenation of data. This format does not have individual file names for each image, but has a reference to the location of the image. NSFBLOB is stored n the NSF header record.

7. X9.100-180 - The Convert to 180 function will read an X9.37 file and generate an output X9.100-180 formatted file in the working directory. *Warning: The X9.100-180 file conforms to the Second Ballot format of X9.100-180. This file is substantially different than previous non balloted reversions of the standard.*

   NOTE: This function requires the X9180 license feature bit.

8. MAL – The Convert to MAL function will read a X9.37 file and generate a FRB Payers Services formatted ASCII file. The title is MAL because the header records are "mal formed" 937 format from a version of a 1994 - X9.37. It is the same as the current X9.37 format but without the image records and a wrong value for the record length field. *NOTE: This function requires the X9180 license feature bit.*

9. UCD – The Convert to UCD function will read a X9.37 file and generate a UCD/187 formatted file.

Appendix C - Record 61 Versions

Overview

X9.37 Does Not Have a Record 61 Defined

The X9.37 DSTU does not have a record 61 to be used for credits or deposit slips. A newer approved standard, X9.100-180 does have such a record. The X9 committee developing the X9.100-180 standard has produced many revisions of proposed standard. Those revisions have been inserted into X9.37 to provide a means for items such as deposit slips. The result is that there is not a single "standard" version of Record 61 in a X9.37 file. Even worse, there is no automatic way to detect which version is being used. In order to print a record 61, any of the variety available needs to be converted to a reasonable choice.

The basic R61 formats are identified with AMP versions. Many special formats are also available depending on the companion documents provided by the receiving bank. Special values are used to determine which version of R61 is being used as input and needed for output. If there is a R61 in the source file, the user must identify the Record 61 format for correct conversion.

The All My Papers IRD Print system uses Version 1 for printing IRDs. For the purposes of printing IRDs, Version 1 is a superset of Version 3. Hence Version 3 record 61s can be used to print with IRD Print without external conversion. The Version 2 of Record 61 is less frequently used. In order to use IRD Print, the AX9LIB function ax9Read937Write937 must be used to convert from the Version 2 to a Version 1

.

Credit Record--AMP Version 1 (First Ballot 180)

1	01-02	2	**61**	Record Type	X9.100-180
2	03-03	1	*Alphanumeric*	Record Usage Indicator	
3	04-18	15	*NBSMD*	Auxiliary On-Us	
4	19-19	1	*Alphanumeric*	External Processing Code Also known as Position 44.	
5	20-28	9	*TTTTAAAAC*	Posting Bank Routing Number Identifies the institution by or through the item is payable. Federal Reserve Routing Symbol represented as *TTTT* ABA Institution Identifier represented as *AAAA* Check Digit represented as *C*	
6	29-48	20	*NBSMOS*	Posting Account Number On-Us Data specified by the payor bank.	
7	49-58	10	*Numeric*	Item Amount US dollar value of the check.	
8	59-73	15	*Numeric*	ECE Institution Item Sequence Number	Assigned by the institution that creates the Check Detail Record
9	74-74	1	**G**	Documentation Type Indicator Indicates the type of documentation that supports the check record.	
10	75-75	1	*Alphanumeric*	Type of Account Code Indicates the credit account type.	
11	76-77	2	*Alphanumeric*	Source of Work Code Identifies the incoming work.	
12	78-80	3	Blank	Reserved	Reserved for future use by the Accredited Standards Committee X9.

Credit Record--AMP Version 2

1	01-02	2	**61**	Record Type	X9.100-180
2	03-17	15	*NBSMD*	Auxiliary On-Us	
3	18-18	1	*Alphanumeric*	External Processing Code Also known as Position 44.	
4	19-27	9	*TTTTAAAAC*	Payor Bank Routing Number Identifies the institution by or through the item is payable. Federal Reserve Routing Symbol represented as *TTTT* ABA Institution Identifier represented as *AAAA* Check Digit represented as *C*	
5	28-47	20	*NBSMOS*	Credit Account Number On-Us Data specified by the payor bank.	
6	48-57	10	*Numeric*	Item Amount US dollar value of the check.	
7	58-72	15	*NB*	ECE Institution Item Sequence Number	Assigned by the institution that creates the Check Detail Record
8	73-73	1	**G**	Documentation Type Indicator Indicates the type of documentation that supports the check record.	X9.100-180
9	74-74	1	*Alphanumeric*	Type of Account Code Indicates the credit account type.	
10	75-75	1	*Alphanumeric*	Source of Work Code Identifies the incoming work.	
11	76-76	1	*Alphanumeric*	Work Type Identifies the type of work.	
12	77-77	1	*Alphanumeric*	Debit/Credit Indicator .	
13	78-80	3	`Blank`	Reserved	Reserved for future use by the Accredited Standards Committee X9.

Credit Record--AMP Version 3

1	01-02	2	**61**	Record Type	X9.100-180
2	03-03	1	*Alphanumeric*	Record Usage Indicator	
3	04-18	15	*NBSMD*	Auxiliary On-Us	
4	19-19	1	*Alphanumeric*	External Processing Code Also known as Position 44.	
5	20-28	9	*TTTTAAAAC*	Posting Bank Routing Number Identifies the institution by or through the item is payable. Federal Reserve Routing Symbol represented as *TTTT* ABA Institution Identifier represented as *AAAA* Check Digit represented as *C*	
6	29-48	20	*NBSMOS*	Posting Account Number On-Us Data specified by the payor bank.	
7	49-58	10	*Numeric*	Item Amount US dollar value of the check.	
8	59-73	15	*Numeric*	Item Sequence Number	Assigned by the institution that creates the Check Detail Record
9	74-74	1	**G**	Documentation Type Indicator Indicates the type of documentation that supports the check record.	
10	75-75	1	*Alphanumeric*	Type of Account Code Indicates the credit account type.	
11	76-76	1	*Alphanumeric*	Source of Work Code Identifies the incoming work.	
12	77-80	4	`Blank`	Reserved	Reserved for future use

Credit Record--Second Ballot 180 (not used in 937)

1	01-02	2	**61**	Record Type	X9.100-180
2	03-03	1	*Alphanumeric*	Record Usage Indicator	
3	04-18	15	*Numeric*	Auxiliary On-Us	
4	19-19	1	*Alphanumeric*	External Processing Code Also known as Position 44.	
5	20-28	9	*TTTTAAAAC*	Payor Bank Routing Number Identifies the institution by or through the item is payable. Federal Reserve Routing Symbol represented as *TTTT* ABA Institution Identifier represented as *AAAA* Check Digit represented as *C*	
6	29-48	20	*Numeric*	Credit Account Number On-Us Data specified by the payor bank.	
7	49-62	14	*Numeric*	Item Amount US dollar value of the check.	
8	63-77	15	*Numeric*	ECE Institution Item Sequence Number	Assigned by the institution that creates the Check Detail Record
9	78-78	1	**G**	Documentation Type Indicator Indicates the type of documentation that supports the check record.	
10	79-79	1	*Alphanumeric*	Type of Account Code Indicates the credit account type.	
11	80-81	2	*Alphanumeric*	Source of Work Code Identifies the incoming work.	
12	82-84	3	Blank	Reserved	Reserved for future use by the Accredited Standards Committee X9.

Credit Record 61 - Format Conversions

Format Conversion - AX9Lib provides parameters to convert to and from the 937 file. For example a 937 file can be converted to a CSV file, and a CSV file can be converted to a 937 file. While AX9LIB supports various formats and specifications, it is not intended to be a converter from one specification to another. If a R61 is used or needs to be created, it is necessary to indicate the credit format being used/needed.

The exact format of Record 61 will depend on the Companion Document provided by the bank and can carry various information in these fields.

AMP's basic Record 61 formatting is controlled by values set in the Source Rec61 Version and Dest Rec61 Version fields.

Many of these codes pertain to the insertion of the various credit formats (R61's or R25's) as they were not originally specified in the 937 specifications. These values are inserted in the SourceRec61Version or DestRec61Version fields depending on function.

In general, any function that reads a 937 file (like ConvertToCSV) requires source61 and source68 values, if there are 61 and/or 68 records. However, in most cases for non-Wells formats, source61 can use the default 1. Note: Generally,Set values equal to the actual version.

See spreadsheet Ax9Lib Formats and Conversion - External V in HELP FILE of AX9LIB for additional information.

Conversion Parameters - Insert Credit Record using the following values in the in the Dest Rec 61 Version depending. **Note**: There are times when this value will be needed in the Source Rec 61 Version depending what R61 format is used in the input file:

DSTU X9.38-2003

1 - Record 61v1 - AMP1

2 - Record 61v2 - AMP2
3 - Record 61v3 - AMP3
5 - Record 61v5 - Wells Fargo - specialty

Base Code: 1-19 = Base Format (1 is Default) - Not **all Base Code formats will accept a Rec61, so no code is available.**

To add a credit Record 61 at Bundle level when the file format is certain of the Base codes 1-19 add +20 to the value of the Base Code. (Most of these formats can be accomplished by using 21, 22, 25.

Example:

DestRec61version = 21 (Base Code (1) + 20 = 21
Will create a Record 61 in AMP1 format at every Bundle level
Values 21, 22, 25

To add a credit Record 61 at Cash Letter level add +40 to the value of the Base Code.
Example:

DestRec61version = 41 (Base Code (1) + 40 = 41
Will create a Record 61 in AMP1 format at Cash Letter level
Value 41, 42, 47

SPECIALTY Formatss:

BOFA:

10 = BOFA 61 Format - when already have a R61, but need to Balance or perform other functions, use =10 in the Dest and Source Fields
30 =Add Credit 61 for every bundle - BOFA
50 = Add Credit 61 for every cashletter - BOFA
BOFA does not include the credit Record 61 totals in the Control Records. When using the Merge Function for BOFA, use Source 61 = 10 or 50 must be included if Record 61s used in input file

Wells Fargo:

25 =Add Credit 61 for every bundle
45 = Add Credit 61 for every cashletter
If modifications are required after the insertion of the R61, special parameters are need:
5 = 936 - if Wells file is a 936
6 = 937 - if Wells file is a 937
Example:
To go from 936 to 937,
DestRec61version = 6 - converting 936 to 937
To go from 937 to 936, the parameters are
DestRec61version = 5 - converting 937 to a 936

Credit 25: (Instead of a Record 61) -Note: Must have front61/back61 images in Working Directory

38 = Add credit 25 for every bundle
58 = Add credit 25 for every cashletter
78

Special:

47 = Add credit record AMP2 format before bundle record
33 = R61 v 180 with 84 characters
82= R61 after Record 70

Appendix D – Record 68 - General Format

X9.100-180 contains a Record 68 User Record which is a conditional record which contains user controlled fields. The record shall be used only under clearing arrangements.

The exact format of Record 68 will depend on the Companion Document provided by the bank and can carry various information in these fields.

AMP's basic Record 68 formatting is controlled by values set in the Source Rec68 Version an Dest Rec68 Version fields. **This applies only if there are Record 68's** in the input file and implies the Source Rec 68 Version (input file) and the Dest Rec 68 Version (output file) will be the same value unless you are converting a Wells 936 to 937 or vice versa, then they will be different. Note: AX9Lib does **not create** a R68, but determines the type and carries it through.

Source Rec 68 Version
or
Dest Rec 68 Version

Property specifies the version of the input Record 68
The exact format of Record 68 will depend on the Companion Document provided by the bank

Valid values include: Set value equal to the actual version

0 = Unknown - Format not specified
1 = Default937 - Variable length
5 = Wells Fargo format (WFED 936 fixed to AMP 937 variable)
6 = Wells Fargo (AMP 937 to WFED 936 - variable to fixed 80)
10 = BOFA (81 Bytes)
14 = Specialty Format (200 bytes)

Appendix E – Error Codes

0	No error
1	An error occurred
4001	NotImplemented - Method is not implemented and not supported yet
4002	ExceptionError - Obsolete - Use NetInternalError instead
4002	NetInternalError - A method threw an exception. Refer to log files for more information.
4003	InputDataTooLong - Length of the input data is too long
4004	ReturnReasonTooLong - Length of the return reason is too long
4005	MissingAx9LiDLL - Ax9lib.dll or Ax9Lib64.dll is not in the path
4006	ReturnReasonInvalid - Return is invalid
4020	WorkingDirectoryNotSet - Working Directory property is not set
4092	NetFileAccessError - File access error
5000	ToolsGeneralError - X9Tools general error
5001	ToolsErrorWithCommandLine - X9Tools error with command line
5002	ToolsInternalError - X9Tools internal exception. Refer to log files for more information
5003	X9Tools error with ini file
5005	ToolsUnknownCommand - Unknown command in command line arguments

1	Could not allocate PC memory space. A local or global allocation failed that was needed to complete the requested operation. (Possibly misspelled function name in Call of AX9Lib)

3	Specified work image does not exist. No image by the given name can be located.
4	Name already in use.
6	Not a primary image. An alias image may not be used in this instance.
10	AX9LIB cannot support any more tasks. The maximum number of callers has already been reached.
11	Internal error. A software error has been detected in the AX9LIB system. Please report this to AllMyPapers Technical Support.

12 Image bounds exceeded. The requested DX, DY, X, Y values exceed the values allowed for this image, as given by MaxHeight and MaxWidth, or the requested sub-image lies outside of the current image dimensions.

13 Image metrics error. The requested sub-image lies outside of the current image dimensions.

14 Internal error calling the Windows API.

15 Bad handle passed to function. The given handle is incorrect or inappropriate for the function in question.

16 User interrupt. A function terminated because of an improper call.

19 AMP function call error. There is an error in the arguments passed to the function in question.

20 No size information. The image has not yet been loaded with any image data and thus has no dimensions.

21 No cross-board operations are allowed. You may not perform an operation where the source and destination image operands reside on different co-processors.

22 Incompatible image sizes. When a destination image is fixed size, the result image must be less than or equal to the size of the destination.

23 Bad file name. The file path name given is incorrect or cannot be opened. Note: could be using different version of DLL. Need trace log from the event for assistance.

24 I/O error. The I/O system reported an error during execution of this function. Note: Could be multiple errors – check log for support.

25 Cannot open trace file.

26 An invalid compression type was given.

27 An internal TIFF operation failed. In the processing of the IFD list or header, some critical operation failed.

28 Required TIFF tag missing. The TIFF 6.0 Specification defines those tags which at a minimum must be present in all baseline TIFF files. One of those tags is missing.

29 Image organization not supported. Only 1 bit per pixel bi-level images are supported.

30 This system is unable to run AX9LIB. Call AllMyPapers Technical Support.

31 Unable to open the requested TIFF file. It may not be a TIFF file, or it has an invalid header.

32 The requested image within a multi-image TIFF file is not in the image file directory of that TIFF file.

33 An error occurred while reading the TIFF IFD.

34 The KDY value given for Group 3 2d compression is invalid.

35 Assertion logic error. Some internal software data or pointer consistency has occurred.

36 No region has been selected to support the requested operation.

37 The number passed to the function is out of range.

40 The resolution value given is not valid.

41 The page size value given is not valid.

42 The operation type is not valid.

43 The mode given is not valid.

46 The scale ratio given is not valid for this operation.

47 One of the arguments passed to the function is invalid.

48 AX9LIB is unable to create the requested file. This is most likely due to an invalid path or some I/O permission error.

49 The margins are not legal for the page size.

51 No file specification was given and is required for this operation.

52 No index string was found in the file path name string.

53 Huge objects not supported yet.

54 The clipboard is empty.

55 General error. No detail available.

56 Download failure.

60 General printer failure.

62 A bad tag was found in a TIFF file.

64 An invalid TIFF header was detected.

65 Scaling while printing requires buffered print mode.

66 Source and destination images must be different.

67 The function in question timed out.

68 A callback function returned an error.

69 Application lockout.

70 This version of AX9LIB is not correct for this application.

71 An invalid file type was specified.

72 The image IX value must be a multiple of 32 for this operation.

73 The margins specified are not legal for this operation.

74 The requested TIFF tag already exists in the IFD list.

75 An invalid Optika header was detected.

76 The requested file format is unsupported in this mode.

83 No ensigns defined or allowed.

84 Bad MODCA RECID parameter

85 IBM MMR format not supported

86 Unsupported compression type

87 Decompression error

88 Unsupported MODCA or IOCA file format

89 Compression error

90 Thread already attached to DLL

91	Disk is full
92	File Access error - Files open in another app
93	Too many files open
94	File exists
95	Bad file handle
96	No such file or directory - if creating Rec. 61, check front/back61 images are in WD
98	Thread not attached
101	No object data in image block
102	Can't find a needed DLL
103	Can't find entry point in DLL
104	License file fails security check
105	License check detected date rollback
106	License expired
107	License required for this feature
108	Image degenerated to dx=0 or dy=0
113	Software implementation only
116	Missing files/files not loaded
131	The MICR result file is missing--usually means fail to verify image and x9.37
132	No image record for at least one check record
135	Source file had invalid records – record number not standard
136	Source file had invalid characters in a non-binary record
137	Problem with Working Directory
138	X9.37 records do not balance
139	X9.37 record not found
140	Reading beyond End of File
141	Field not wide enough for parameter
142	TOC has zero entries
143	File Dup found
144	EOF because of corrupted file
145	EOF because of truncated file - Bad or corrupted file
146	Not valid X9.37 -- cannot determine byte order/character type
147	Record number invalid (usually a parameter) –it may be Image Reference Key length in Record 51. Set value to 0
148	Function incomplete or incorrect
149	NSF file not well formed
150	Cannot Print -- See the log file for the results of Print Ready Test

151 Does Not Conform -- See the log file for the results of Print Ready Test – I**nformational message – still creates 937 – images are wrong and up to user to determine if it is an error in their particular case**

152 Does Not Balance -- See the log file for the results of Print Ready Test - **Informational – still creates 937 – images are wrong and up to user to determine if it is an error in their particular case**

153 MICR Batch missing – Informational – still creates 937 – images wrong and up to user to determine if it is an error in their particular case

154 Selected IRDs where not printed based on MICR Verify result.

155 Source file had invalid records.--record number not standard

156 Source file had invalid characters in a non-binary record – The source file is improperly formed

157 AmpLib is missing or out of date

158 Translating Table error going to/from NSF

159 Too many items for license – License Exceeded

160 File too large for page swap

161 Does not match type

162 CSV format error

163 I/O error using MM

164 Too Few Files Specified - 2 or more files are required in the list file.

165 Invalid68Record Error reading record 68 most likely the wrong 68 version was used

166 DestinationFile Not Set - A destination file is required and has not been set

167 Wells File Required - Input file must be Wells (936)

168 Wells File Not Allowed - Input file cannot be Wells (936)

169 Invalid Field Type - Obsolete - Use Invalid Arg Field Type instead

169 Invalid ArgField Type - Invalid argument for field type.

170 InvalidInputData - Obsolete. Us Invalid ArgInputData instead

171 InvalidArgRouting - CreatorRouting must be populated

172 InvalidArgCreatorDate - Creator Date must be populated

173 InvalidFieldWidthInputData - Invalid field width for Input Data

174 InvalidFieldWidthSequence - Invalid field width for Sequence

Appendix F – Sample Reports/Logs

XML Report:

This format is generated if the Report File field is filled.

Example of XML Report for Exchange Ready processing:


```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="X9QualifierReport.xsl"?>
<transmittal xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<FileSource>
<FileParameters>
<FileName>C:\Test Files - 937\temp (2).nsf.937</FileName>
<TestSoftware>All My Papers Exchange Ready</TestSoftware>
<TestVersion>All My Papers X9Lib 6.2.9.0</TestVersion>
<TestLevel>2</TestLevel>
<StandardLevel>3</StandardLevel>
<ControlRecords>Balanced</ControlRecords>
<FileItems>2</FileItems>
<FileAmount>1500000</FileAmount>
<FileDate>20140614</FileDate>
<FileTime>1221</FileTime>
<OriginRouting>987654320</OriginRouting>
<OriginName>              </OriginName>
<DestinationRouting>123456780</DestinationRouting>
<DestinationName>              </DestinationName>
<CharType>EBCDIC</CharType>
<ByteOrder>Motorola</ByteOrder>
</FileParameters>
</FileSource>
</transmittal>
```

Log Report for Exchange Ready:

The format of this report is identical for all functions and the information displayed depends on the function performed. It is always placed in the Working Directory using the original Source File name. Open with Notepad.

C:\tempdemotest\temp.nsf.937

```
All My Papers Log File for C:\WinSaige\DevAMP\TestFiles\x937\Test3Items.937
WARNING -- x9.37 DSTU March 31, 2003
Compile Aug  6 2014  13:01:46   Mod Tue Feb 25 17:01:06 2014
Date 08/11/14 Time 11:44:12

Function ax9PrintReady_sub

Application Directory       C:\WinSaige\DevAMP\SVN\ax9lib\bin\Debug\
Working    Directory       c:\work\
Destination File          (Null Str)
Aux        File           (Null Ptr)
Report     File           (Null Str)
Pick       File           (Null Str)

All My Papers X9Lib 6.3.0.992   Item Count Authorized  10000000
Parameters:
Convert Images  0 Repair Images 0 Repair Threshold 6 Repair Size 140000
Bundle Limit    0 Fed Format    0
Record 61 In    1 Record 61 Out 0
Record 68 In    1 Record 68 Out 0
Cash Letter     0 Bundle       0 Item          0  Record     0 Field 0 Occurance 0
Data Length 3072
Account                 (Null Str)
Sequence                (Null Str)
Cash Letter ID          (Null Str)
Creator Date            (Null Str)
Creator Routing         (Null Str)
Destination Routing     (Null Str)
Data                    (Null Str)
Tough Tiff Parameters:
Max Heigth              4.800000  Max Width              9.400000   Min Heigth       2.200000   Min Width        5.500000  Decomp Test 0
Max Heigth Personal     0.000000  Max Width Personal     0.000000   Min Heigth Personal 0.000000   Min Width Personal 0.000000
Decomp Test 0
Front Back Max Diff Heigth 0.600000  Front Back Max Diff Width  0.500000 Treat non ABA as Personal 0
Front Max Darkness      39.000000 Front Min Darkness      2.100000  Back Max Darkness 39.000000  Back Min Darkness 0.000000
Front Skew              2.700000 Back Skew              360.000000
Front UL                1.000000 Front LL               0.800000  Front UR         1.000000 Front LR          1.000000
Front Noise             500  Back Noise             500   Front Spot Width     3   Front Spot Height    3
Aux onus Min Conf           0  Route Min Conf            94   Onus Min Conf       40   Amount Min Conf      0
Process all items           0  Process upside down       0   Min Confidence       0
Analysis File (null)
ExchangeEx Mask c4bddffb
ReadyTest Type 2  ExchangeEx Mask Corrected  c4bddffb Server 0
 All My Papers, A California Corporation AmpLib 6, 2, 8, 16
Items in file 3 Pool Size 0

**PrintReady** Cash Letter Index  1 ID ADVO4822 Bundle Index     1 ID 1144822

Processed Bundle 1144822
Processed Bundle Dollar Total 800 Item Total 3
Processed Bundle NoPrint Item Total 0
Record    Bundle Dollar Total 800 Item Total 3
Processed Cash Letter ID ADVO4822
```

Processed Cash Letter Dollar Total 800 Item Total 3 Bundles 1
Record Cash Letter Dollar Total 800 Item Total 3 Bundles 1

Processed File Dollar Amount 800 Item Total 3 Cash Letters 1
Record File Dollar Amount 800 Item Total 3 Cash Letters 1 Record Total 33

All My Papers--x9.37 Record Frequency Report
Record 1 Frequency 1
Record 10 Frequency 1
Record 20 Frequency 1
Record 25 Frequency 3
Record 26 Frequency 3
Record 28 Frequency 3
Record 50 Frequency 6
Record 52 Frequency 6
Record 54 Frequency 6
Record 70 Frequency 1
Record 90 Frequency 1
Record 99 Frequency 1
File: C:\WinSaige\DevAMP\TestFiles\x937\Test3Items.937
 Dollars: 8.00 Items: 3 Cash Letters: 1 Records: 33 Credits: 0 Images: 6
All My Papers--X9.100-181 Item Conformance Summary as Determined by Test Selected
    3 Conforming Items
    0 Non Conforming Items
Non Conforming Image Detail--May be multiple issues per item
Occurances Tested
    0 1 TQA Not TIFF -- Not a TIFF Wrapper
    0 1 TQA Multi Strip -- Single Strip Norm
    0 1 TQA Non Group 4 -- Group 4 2d Compression Norm
    0 1 TQA X Res Not 200/240 or Front/Back Do Not Match -- 200/240 DPI Norm
    0 1 TQA Y Res Not 200/240 or Front/Back Do Not Match -- 200/240 DPI Norm
    0 1 TQA Resolution Not Set -- Cannot be zero
    0 1 TQA Not Intel -- TIFF Byte Order Little Endian Norm
    0 1 TQA Fill Order Not 1 -- Byte Fill Order 1 Norm
    0 1 TQA NewSubFileType Not 0 or Multi Images -- Single Image TIFF File Norm
    0 1 TQA BitsPerSample Not 1 -- Black/White Norm
    0 1 TQA Orientation Not 1 -- Upright Norm
    0 1 TQA Photometric Interpretation Not 0
    0 1 TQA Missing Tags -- Required Tag Missing
    0 1 TQA Missing Images -- Image Missing
    0 1 TQA Data Type Error -- Data Type or Invalid Count Found
    0 1 TQA Options Set Error -- Option set(293) and not default value or invalid option(292)
    0 1 TQA EOFB Error -- EOFB missing or too much fill
    0 0 TQA Private/ASCII Tags -- Discretionary exclusion
    0 1 TQA General -- General TIFF (lengths, tag order, tag value, tag dup)

    0 0 IQA -- Dimension Out of Bounds
    0 0 IQA -- Compressed Size Out of Bounds
    0 0 IQA -- Decompression Failure
    0 0 IQA -- Darkness Out of Bounds
    0 0 IQA -- Skew Out of Bounds
    0 0 IQA -- Noise Out of Bounds
    0 0 IQA -- Turned Corners

    0 0 MQA -- MICR MisMatch
    0 0 MQA -- Asterisk in 937 data

    0 1 FQA -- File UnBalanced
    0 0 FQA -- File Type Does Not Match
    0 1 FQA -- Bad Characters Found (Record Count)
Source file contains 1 different image sources(scanner vendor combinations). Source file contains 0 IRD.
Start Date 08/11/14 Time 11:44:12 -- Stop Date 08/11/14 Time 11:44:14 Source Conformance Status 0

Appendix G – Test Exchange Error Definitions

Screen Display – Test Exchange Ready Error Report Definitions

All items tested are on the Summary Test Exchange Ready Error Report displayed on the screen after the function is executed. Below are definitions of what the various fields on the report indicate:

Multi Strip – Single Strip Norm:
Strips of data – in the "old days", many strips (lines per image) with different strip lengths were standard. That is no longer allowed. All image views must be represented as a single strip. Multi-strips will NOT conform to Tough TIFF

Non Group 4 – Group 4 2d Compression Norm:
Different types of compression algorithms. The only thing that is allowed in X937 format is Group 4 2D.

X Res Not 200/240 – 200/240 DPI Norm:
200/240 is the only resolution allowed. If it is not this resolution, it will not work.

Y Res Not 200/240 – 200/240 DPI Norm:
200/240 is the only resolution allowed. It must be this resolution and must be the same as the X Res or it will not work.

Resolution Not Set - Cannot be zero

Tiff Not Intel – TIFF Byte Order Little Endian Norm:
Motorola (Little Endian) is the only byte order allowed for Tough TIFF. It can not be Intel.

Fill Order not 1 - Byte Fill Order 1 Norm: (Should be Bit order) – left or rightmost bit in word. Only 1 is acceptable.

New SubFile Type Not 0 – Single Image TIFF File Norm:
Previously, 1 file could contain many TIFF images. This is NO longer acceptable. Tough TIFF needs 1 image per file – each front and back must be separate.

BitsperSample Not 1 – Black/White Norm:
Binary – Black/white opposed to color or gray scale. Must NOT

be gray scale or color. (Canada has separate set of problems here.)

Orientation Not 1- Upright Norm:
TIFF allows different ways to present the page on the screen – Can be flipped or rotated at display time. This is NO longer acceptable. For Tough TIFF, the image must be upright (but can have a variance – we'll discuss later.)

Photometric Interpretation Not 0

Missing Tags - Required - Tag Missing - These are the resolution tags and without them it is not possible to determine how large the image is supposed to be for Tough Tiff.

\ Missing Images - Required - Image Missing

Data Type Error - Data Type or invalid count found
Options Set error - Options set(293) and not default value or invalid option (292) - Need certain tags or set to default value

EOFB Error - EOFB missing or too much fill

General - General TIFF (Lengths, Tag Order, Tag Value, Tag dup)
Misc. errors – i.e., Tiff tags conflict with similar information in the X9.37 file (such as the data length and Tiff length tag indicating how long the image is supposed to be). If lengths don't match, we flag as bad item.
EOFB facsimile block code needs to be present. Previously, it was ignored because it was usually wrong. Now, they will be flagged as an error, but can convert these to conforming.

Note: IQA parameters are not for Tough Tiff, but specified by AMP

IQA Quick – Dimension Out of Bounds:
Used to test Tiff files which are received from home scanners and are converted to black/white through AMPLib
Dimension of image.
IQA – Compressed Size Out of Bounds:
Image too big and is not a valid check. This parameter can also detect a piggy-back. Concerns physical dimensions.
IQA - Decompression Failure
IQA - Darkness Out of Bounds
IQA - Skew Out of Bounds
IQA - Noise Out of Bounds
IQA - Turned Corners
MQA - MICR Mismatch
MQA - Asterisk in 937 data
FQA - File unbalanced
FQA - File Type does not match
FAQ - Bad Characters found (record count)

Error 150 means cannot print, but can decide what to do and Run Test Exchange Ready.

Appendix H – .NET Interface Commands for accessing AX9LIB

Partial list of interface commands. A complete list of AX9LIB Net object can be found in the AX9LIBNET.CHM file in the Help folder of Ax9lib. Capitalization and spelling must be correct for AX9Lib to execute.

Enumeration	Description
X9ByteOrder	Byte order for the X9 file
X9CharType	Character encoding type for the X9 file
X9CorrectType	Correctable type
X9EndorseType	Endorsement type
X9ProgressStatus	Used to cancel a long process
X9Record68Format	Record 68 format type
X9Result	Return results from X9 methods
X9SortOrder	Indicates how to sort the X9 file
X9VerifyMode	Used for IRD mode

Name	Description
Append937	Appends X9.37 Items to an X9.37 file. Changes the source X9.37 file by appending items located in a separate X9.37 format file to the Item position of the source file.
AppendCsv	Appends an item to a csv file
BadCharCorrection	Tests for bad characters and creates a report. If a destination file is specified, corrects bad characters and generates a new file with the corrected characters.
Balance	Rebalances the file
ConvertFrom937	Reads an X9.37 file and outputs an X9.37 file
ConvertFrom937Ex	Deletes records during the 937 conversion.
ConvertFromCsv	Converts a CSV file to a 937 file
ConvertFromImages	Convert a folder of images to a 937 file. Not implemented.
ConvertFromMal	Converts from a MAL file to a 937 file
ConvertFromNsf	Reads an NSF file and outputs an X9.37 file. Reads an Normalized Scanner Format (NSF) file combined with Single Image TIFF (SIT) files and output an X9.37 file.
ConvertFromUCD	Converts a UCD file to a standard X9.37 file. and the input 937 file szFileName and output an X9.37 file in the working directory with the name szFileName.937.
ConvertTo180	Converts a 937 file to a 180 file
ConvertToCsvBlob	Converts a 937 file to a CSV BLOB file

ConvertToCsvSit	Converts a 937 file to a CSV SIT file
ConvertToMal	Converts from a 937 file to a MAL file
ConvertToNsfBlob	Converts a 937 to a NSF BLOB file
ConvertToNsfMit	**Obsolete.** Converts from a 937 file to a NSF MIT file
ConvertToNsfSit	Converts a 937 file to a NSF SIT file
ConvertToSOP	Converts a 937 file to SOP
ConvertToUCD	Converts a standard X9.37 file to a UCD file
CreateNotCorrectableList	Creates a new picklist file that only has the 937 items that are not corrrectable in a ConvertFrom937 operation
CreateNotCorrectableListEx	Creates a new picklist file for a correction type that only has the 937 items that are not corrrectable in a ConvertFrom937 operation
CutterSort	Sorts the records of a 937 file. This method is used to sort X9.37 files in preparation for IRD Printing. The fileName is sorted so that the output can placed on a multiple page cutter and all the pages cut at once. In order that the resulting stacks of IRDs can be bunched together, multiple pages will have VOID checks inserted as needed.
EnableTrace	**Obsolete.** Use EnableVerboseLogging instead
Endorse937	Adds an endorsement to the rear check images in the X.937 file.
FindPick	Creates a pick list by searching for matching

	data in the specified field
GetAmpLibVersion	Gets the current version of AmpLib.dll or AmpLib64.dll
GetCOMVersion	**Obsolete.** Use GetVersion instead
GetField	Retrieves data from the specified fields of an X9.37 file.
GetFieldTOC	Retrieves data from the specified fields of an X9.37 file using an already TOC file.
GetFileType	Returns the file attributes of the 937 file
GetHeaderInFile	Retreives the header information from a file
GetItemsInFile	Reads the last record in the file to determin the item counts
GetLicense	Get license information from the AX937LIB license
GetLicenseAvailability	Indicates if a license is available or not.
GetLicenseEx	Get license information from the AX937LIB and AMPLIB license
GetLicenseInfo	Get all license information from the AX937LIB license
GetMessage	Retrieves the message text for the last result.
GetVersion	Gets the current version of Ax9Lib.dll or Ax9Lib64.dll
GroupAccount	Groups items into multiple files by account

ItemCopy	Copies item
ItemDelete	Delete a specific item
LimitBundles	Limits the bundle size during 9.37 conversion.
ListCopy	Copies multiple items
ListDelete	Deletes items contained in the list file/aux file.
LoadTOC	Loads a TOC (Table of Contents) file into memory for optimal performance
LogErrorMessage	Logs an error message to the log file and to ax9lib.err
LogMessage	Logs a message to the log file.
Merge	Combines a list of files. Changes the source X9.37 file by prepending items located in a separate (and partial) X9.37 format file before the item position of the source file.
MergeBundles	Reads an X9.37 file and outputs an X9.37 file merging multiple Bundles sections into a single bundle during the process.
MergeCashLetters	Reads an X9.37 file and outputs an X9.37 file merging multiple Cash Letter sections into a single Cash Letter during the process.
MergeFiles(String, Boolean, Boolean)	Combines a list of 9.37 files.
MergeFiles(String, X9CharType, X9ByteOrder, Boolean, Boolean)	**Obsolete.** Combines a list of 9.37 files.
MicrParse	Parses a micr line from a string

OutputTraceMessage	**Obsolete.** Use LogMessage() instead. Write a message to the trace file, which traces detailed information for the Ax9Lib native dll
ParseAccount	Parse the account information from the MICR line
Prepend937	Prepends an item to a 937 file
PrependCsv	Prepends an item to a csv file
SecurityEnableAppsFile	Enables an OEM license file
SelectItem937	Extracts an item from a 937 file
SelectItemCsv	Extracts an item from a CSV file
SelectList937	Extracts a list of items from a 937 file
SelectListCsv	Extracts a list of items from a CSV file
SetAllFields	Sets all occurrences of a field in an X9.37 file. Writes data to every occurrence of the specified field in an X9.37 file.
SetEndorseDimensions	Set endorsement dimensions
SetEndorseOptions	Set endorsement options
SetField	Writes data to the specified field of an X9.37 file.
Sort	Used in conjunction with Merge to group items by Payor Routing (Record 25, Field 4). Creates an intermediate file for each Routing number in the working directory. Also creates AMP.TOF containing a list of these intermediate files.

SortBofd	Used in conjunction with Merge to group items by Bofd Routing (Record 26, Field 3). Creates an intermediate file for each Bofd Routing number in the working directory. Also creates AMP.TOF containing a list of these intermediate files.
SortBofdAccount	Used in conjunction with Merge to group items by Bofd Account (Record 26, Field 6). Creates an intermediate file for each Bofd Routing number in the working directory. Also creates AMP.TOF containing a list of these intermediate files.
SortDetectDups	The SortDetectDups method performs a sort operation on an X9.37 file to identify duplicate items in the file. It is a quick way to determine if a check has been submitted multiple times.
SortItems	Sorts file by record 25 or by record 31 for numeric fields only
SortMicrFields	Extracts items based on MICR fields
SortPayor	Same as Sort except can pass in Auxiliary file. Used in conjunction with Merge to group items by Payor Routing (Record 25, Field 4). Creates an intermediate file for each Bofd Routing number in the working directory. Also creates AMP.TOF containing a list of these intermediate files.
Split	Splits a file
SplitCashLetter	Splits a cash letter in a file
StartProgressDialog	**Obsolete.** Use ShowProgressDialog=true and/or ProgressCallback instead

StartProgressDialogEx	**Obsolete.** Use ShowProgressDialog=true and/or ProgressCallback instead
StopProgressDialog	**Obsolete.** Use ShowProgressDialog=false and/or ProgressCallback instead
StopProgressDialogEx	**Obsolete.** Use ShowProgressDialog=false and/or ProgressCallback instead
Template	Creates a 937 file with no items
TestAddDup	Adds entry to a file history
TestBalance	Tests if the file is balanced
TestDebug	Not implemented
TestDup	Used to query a file history. The input file, fileName, is compared with a history file containing the "fingerprint" data for all the previously tested files. If the file does not match any of the previous files, the result is zero. If there is a match, the result is non-zero and the name of the duplicate is returned in data.
TestExchangeReady	Verifies that a file is properly formed for general X9.37 exchange
TestPrintReady	Verifies that a file is ready to print IRDs
UnLoadTOC	Unloads TOC (Table of Contents) file from memory.
UpdateProgressDialog	**Obsolete.** Use ShowProgressDialog=true and/or

	ProgressCallback instead
Verify937ToNsfBlob	Requires MicrBatch.exe, MicrBatch64.exe and MicrBatch.ini
WriteReturn	Creates a return file from a forward file. It uses the listFileName/auxFileName to determine what items and return return reason codes to include in the new return file.
WriteReturnAll	Creates a return file from an X9.37 source file, and writes all of the items using using the same Return Reason Code.
WriteReturnFindList	Creates a return file from an X9.37 source file and a list file of items to match.
WriteReturnItem	Creates a return file from a single item. Finds an item in a forward file (record 25), converts it to a return item (record 31), and writes a return file.
WriteReturnToForward	Creates are foward file from a return file. Creates a X9.37 file from a return file and a list file of items to extract.
WriteReturnToForwardAll	**Obsolete.** Use WriteReturnAll instead
WriteReturnToForwardFindList	**Obsolete.** Use WriteReturnFindList instead
WriteReturnToForwardItem	Creates a return file from a single item. Finds and item in a return file (record 31), coverts it to a forward item (record 25), and writes a forward file.