

X9Ware Installation Guide

X9Ware

Your x9.37+ACH+CPA005 support tools

Revision Date: 04/18/2024

Release R5.04

Copyright 2012 – 2022 X9Ware LLC

All enclosed information is proprietary to X9Ware LLC

X9Ware LLC

10753 Indian Head Industrial Blvd

St Louis, Missouri 63132-1101

(844) 937-1850

Email: sales@x9ware.com

Table of Contents

Overview.....	4
Installing X9Assist and X9Utilities.....	11
Minimum Requirements.....	11
Java Heap size when using our Installation Packages.....	11
X9Assist Installation Packages for R4.10 and Higher.....	11
X9Assist Installation Packages for R4.09 and Earlier.....	12
X9Utilities Installation Packages.....	12
Installation Overview.....	13
Installation Notes.....	13
Running X9Utilities in the Cloud.....	14
Availability of New Releases.....	14
X9Assist Failed to Start.....	15
X9Assist Windows Stack Trace.....	16
X9Assist Installation on Linux and macOS.....	17
X9Assist Installation on Linux.....	17
X9Assist Installation on macOS.....	18
X9Assist Docker Containers.....	18
X9Assist Automated File Copies.....	22
MSI Packages and Silent Operations.....	23
Runtime Folders.....	25
Launching X9Assist.....	26
Location of the User Home Folder.....	26
Running From our Native Windows EXE.....	27
Running Within a Java JVM.....	27
Running Within a Java JVM on Windows.....	27
Sample JVM Batch Script for Windows.....	28
License Key Management.....	29
License Key Formats.....	29
X9Assist License Key Registration.....	29
X9Utilities / X9Export License Keys.....	31
X9Ware SDK / E13B-OCR License Keys.....	31
License Key Considerations.....	32
License Key Deployment.....	32
Embedding a License in X9Utilities JAR.....	33
License History.....	34
MICR Fonts.....	35
MICR Font XML definitions.....	35
EOFB.....	38
X9 Standards.....	39
What does EOFB look like?.....	39
Validation Process.....	40
How can your x9 tool provide good EOFB support?.....	40
EOFB Logging.....	40

X9Assist Validation and Support.....	41
Summary.....	42
X9Assist Panel Sizes.....	43
Error Messages.....	44
Message Editor.....	44
Error Message Patterns.....	44
Error Severities.....	45
Customization.....	46
Validation of 4x4 Routings.....	46
Missing Image Documents.....	46
Credit Reconciliation Documents.....	47
Rules and Configurations.....	48
Properties File.....	50
Properties File Location.....	50
Properties File Content.....	50
Properties File Sample.....	53
Properties File Example.....	53
User Profiles.....	54
LDAP (Active Directory) Assignments.....	54
User Permissions Xml.....	55
User Permissions Skeleton.....	56
X9Assist Login Panel.....	56
LDAP Parameters.....	56
User Profiles Summary.....	57
Windows File Extension Association.....	58
CSV File Requirements.....	59
Editing CSV Files.....	59
Other CSV Tools.....	59
Comparing CSV Files.....	60
MICR Printers.....	61
Large Monitor / HD Screen Support.....	63
Server and Citrix Support.....	65
CPA 015 Support.....	67
X9.37 Type 24-34 Image Archive Record Support.....	72
X9.37 Type 61-62 Credit Record Support.....	74
X9.37 Type 68 User Record Support.....	79
X9.37 Extended File Editing.....	89
ACH Configurations.....	90
Customization of Nacha-Site-Validations.....	90
Default Nacha-Site-Validations XML File.....	91
Branding.....	93
FAQ.....	94

Overview

X9Ware LLC offers a product line that includes a wide variety of support tools that improve productivity levels when working with the X9, ACH, and CPA005 file formats. These tools provide a logical progression of capabilities, from our viewer tool through a full toolkit with extensive file validation, manipulation, and creation capabilities. Please refer to our Product Brochures for summaries and our Product Schedule for pricing. Our products have extensive application functionality in the following areas:

1) X9 Files

- Character sets can be defined as EBCDIC, ASCII, or either.
- Field zero record lengths can be defined as required, optional, or either.
- Acceptable tiff image formats can be defined as little endian, big endian, or either.
- Maximum limits can be defined for checks in a bundle, checks in a cash letter, checks in a file, and file size in megabytes.
- Minimum dates can be defined in support of date validation.
- Validation support for X9.100-187, X9.100-187+UCD, X9.100-180, and CPA-015 including numerous common variants for these standards.
- Support for industry defined type 61 credit reconciliation record formats (DSTU and Metavante) including the ability to dynamically determine which format is present and validate accordingly. Credits are automatically balanced against debits.
- Support for the industry defined type 62 credit record per X9.100-187-2013.
- Support for the type 68 user record including the payee endorsement record and any user defined record types that follow standard formatting conventions. Type 68 records can be defined as standard fields or attached as an XML document that is embedded within the type 68 user record.
- Internally efficient with support for very large files.

2) ACH Files

- Character sets can be defined as EBCDIC, ASCII, or either.
- All standard entry classes are supported.
- All addenda types are supported.
- Minimum dates can be defined in support of date validation.
- Internally efficient with support for very large files.
- All X9Assist tools are available including find, validate, modify, search/replace, export, merge, make/generate, scrub, repair, create, filtering, Excel export, compare, etc.

3) CPA005 Files

- 1) Character sets can be defined as EBCDIC, ASCII, or either.
- 2) Transaction files (1464 files) are supported including debits, credits, reversals, and returns.
- 3) Change Notices (208 files) are support.
- 4) Minimum dates can be defined in support of date validation.
- 5) Internally efficient with support for very large files.
- 6) All X9Assist tools are available including find, validate, modify, search/replace, export, merge, make/generate, scrub, repair, create, filtering, Excel export, compare, etc.

4) File Validation (available for all file formats)

- Record and field level editing are performed per the selected rules.

- Errors are reported in detail and can be sorted on a variety of fields.
 - The viewer can be launched to any error from the error list through a single click.
 - Errors are summarized by type with a summary report that shows which errors occur with what frequency within the file.
 - Each error has a corresponding severity level (error, warn, info) that can be tailored as needed by the customer through XML configuration changes.
- 5) Custom file formats and edits (available for all file formats)
- Field validation rules can be modified through the XML configuration. This allows the addition of custom edits based on variant or specialized requirements.
 - Message severity can be changed using XML rules (Error, Warn, Info, or Ignore).
 - Fields can be added to existing records in support of file variants without the needs for other code changes. A user or reserve field can be broken up into multiple fields which can be defined with their presence (mandatory or conditional), justification (left or right), editing rules, specifically required values, and so forth. These fields are displayed in the viewer based on field name, just like any other record field.
 - Records can be added to the XML configuration file, allowing variant requirements to be met without the need for other code changes.
- 6) Tiff Image Validation (X9 only)
- X9.100-181 rules are predefined and can be applied per configuration rules.
 - Ability to customize required tiff tags and values per XML configuration files without X9Assist code changes.
 - EOFB validations are applied per XML configuration rules.
 - Minimum and maximum image sizes are validated per XML configuration rules.
- 7) Viewers (available for all file formats)
- Field viewer shows each record on a field by field basis which includes the record type, field number, field name, field value, position, length, and various field level attributes that simplifies decoding and understanding the contents within the file.
 - Record Viewer shows each data record format and includes all validation errors.
 - Item Viewer has multiple panels where the default view shows the records and data fields for each item with their associated front and back images.
 - For X9 files, there are views that include large scale front and back images and the tiff tags that are associated with each image.
 - A hex viewer is included which allow file content to be viewed as present within the input file.
 - Filter Viewer is similar to the Item Viewer but allows filter items to be viewed, browsed, and individually removed from the filter.
- 8) Tiff Images (X9 only)
- Front and back images can be viewed for each item.
 - Images can be rotated for assistance when they were captured as flipped.
 - To assist in readability, images can be despeckled to remove background noise.
 - Images can be exported to an output folder on a Tiff “as-is” basis, to allow you to get an exact copy of the image as it exists on the X9 file.
 - Images can be copied to the Windows clipboard, allowing them to be easily utilized by other tools of your choosing.
- 9) Find/Filter (available for all file formats)
- Find checks based on field level values.

- Create a filter based on field level values.
 - Find/filter can be done based on: amount range; sequence number range; account account; check number; and within user defined record types and fields.
- 10) Search/Replace (available for all file formats)
- Search for strings at the field level within record type.
 - Search strings can be by value, range, leading, trailing, contains, blanks, or RegEx.
 - Replacement strings can be entered and then applied as directed.
 - All changes are included in the modification log which can be exported by Save.
- 11) Reporting (available for all file formats)
- Fields by record.
 - Error summary.
 - Error detail.
 - Tiff tags on each image (X9).
 - Tiff summary of all tags present across all images (X9).
 - Cash letters (X9).
 - Batches / Bundles.
 - Items.
- 12) Make (available for all file formats)
- Make creates items from either your use cases or from randomly created use cases.
 - Make allows the definition of “reformatter rules” that transform your use case definitions into the format that is needed by Generate.
 - You can save your reformatter rules for repetitive use.
 - Your use case files can be defined and maintained in MS-Excel.
 - You can extract use cases from your test bed and then use them in Make.
 - Various fields (account, amounts, date, payee, memo, etc) can be taken from your use cases or can be assigned via the Make facility.
 - The use case editor can be used to create test cases within your desired account number ranges using your check digit routines.
- 13) Generate (available for all file formats)
- Generate creates output files from the items than are manufactured by Make.
 - Generate provides complete control over the headers, trailers, and batching.
 - Generated test files not only target specific use cases, but can also resolve confidentiality issues associated with the use of production files in your test environments.
 - Generate can be optionally invoked (in addition to the Make interface) using CSV files of your creation.
 - For X9 files, images can be dynamically drawn from the item level use case data. This process allows you to generate test files where the image matches the type 25/31 records. Image creation can be done using varying check artwork and fonts.
 - File header contents can be specified (origination, destination, test vs. prod, etc).
 - Cash letter contents can be specified (origination, destination, collection type, record type, Fed work type, etc).
 - Addendas can be optionally created based on provided parameters.
 - Generated files are run against all field level validations.
 - Generated files can be saved in any industry standard file format.
- 14) Modify (available for all file formats)
- Ability to change individual fields.

- All field level edits are applied with the ability to override if you want to purposely assign invalid values.
- Changes are made to an “in memory” version of the file and not externally applied until you actually create a new output file.
- As many changes can be keyed as desired.
- The modification log tracks all changes that have been made and shows the original value along with the new value that has been assigned. The log can be exported as an audit trail and represents the final changes that were made to the file.
- Individual changes can be reverted to their original values and the log is updated accordingly.
- Validation can be run repeatedly and at any time, which performs cross record validations per the rules definition.
- For X9 files there is the ability to replace individual images with either a standard missing document or any other external tiff image.
- Control (trailer) totals can be automatically updated as individual modifications are applied. All control totals updates are included in the modification log.
- A new output file can be created after changes have been made using the save facility. This new file can have different attributes than the original input file (eg, EBCDIC versus ASCII encoding can be changed).

15) Repair (available for all file formats)

- Repair is an automated feature which allows common field level problems to be automatically fixed without having to manually update the file.
- Repair allows you to select the specific record types that are eligible for repair.
- Repairs are applied with the meaning of the field data can be inferred from the data value that is present. For example, fields that are incorrectly justified can be corrected; trailer totals can be adjusted per the actual items; reserved fields can be altered to blanks when populated; addenda counts and numbering can be updated from the actual addenda present; x9 images can be repaired.
- Summary statistics are provided on the number of repair actions by record type.
- All repaired fields (with before and after values) are included in the modification log and can be exported by Save.
- As with any file repair operation, this function must be carefully used with the results fully validated by the client.

16) Scrub (available for all file formats)

- Creates a revised file by sanitizing proprietary and confidential information.
- Scrub is very useful when you need to create a test file from a production file, or when you need to provide a sample file to a third party but you must first remove your customer confidential information.
- Scrub provides discrete control over the individual fields that will be updated.
- Scrub typically replaces data with randomized information to obscure content.
- Account and ABA numbers are a special case, where they can be replaced with values per your requirements (account number range, check digit routine, etc).
- X9 images are scrubbed by dynamically drawing a replacement check image using the assigned field level attributes of the item. For example, the scrubbed check can have a new amount, account, ABA, or check serial number. A new replacement image is drawn with

this information to match the revised type 25 check detail record. There is absolutely no reference to the original front or back image.

17) Import (available for all file formats)

- Files can be created from your created CSV files.
- Input CSV character strings can be enclosed in single or double quotes.
- Incoming CSV files can be encoded in either the ASCII or EBCDIC character sets.
- Trailer totals can be present on the incoming CSV or automatically calculated.
- For X9, check images can be provided for each item in a separate input folder, and they will be included in the type 52 record that is created.
- For X9, check images can alternatively be dynamically drawn using information from the type 25 and type 31 records (and optionally also for the type 61 and 62 credit records).
- For X9, A standard or customized “missing” document can be used as an image replacement; the generated missing document can optionally include a formatted MICR line.

18) Export (available for all file formats)

- Export output can be CSV, text, or XML.
- A file that is exported and then subsequently imported will have the same content as the original file.
- X9 images can be optionally written to an output folder.
- Multi-file Export will export the contents of a large number of user selected files in a single export operation, to a single output CSV file. Input file names can be inserted into the output CSV file to provide association back to the input data.

19) MICR Print (X9 only)

- Item selection through our filter lists or via the entire file.
- Print format can be customized in terms of page layout and borders.
- Print can be either one or two sided and optionally collated.
- Zoom can be used to review image pages prior to printing.
- Results can be routed to a printer of your choice or zipped to an output file.

20) IRD Print (X9 Only)

- Items can be selected individually or by bundle.
- Routing, sequence number, return reason can be specific for the IRD creator.
- Selection list can be reviewed prior to printing.
- Print function are implemented through our MICR Print capabilities.

21) Compare (available for all file formats)

- Synchronized compare by item amount of two files as selected by the user.
- Compare of both record.
- Images are compared for X9 files.
- Differences are reported at field level showing logical values on both sides.
- Resulting differences text file can be saved for reference and distribution.
- Optional interface to an external comparison tool for visual matching and drill down; batch file is provided for WinMerge as a sample of this implementation.

22) Repackage (available for all file formats)

- Allows the currently loaded file to be re-bundled with a user defined number of items inserted into each bundle.
- Bundle attributes (origination routing, destination routing, business date, creation date, collection indicator, cycle number) can be set with defaults taken from the first bundle.

- Items can be optionally reordered by amount or item sequence number.
- 23) Tiff Tester (X9 Only)
- TIFF image validation can be applied against a single TIFF image located in an external file within the file system (not embedded within an X9 file).
 - TIFF tags for the image are displayed.
 - All X9Assist standard validations are applied and any identified errors are reported.
- 24) Duplicate Item Detection (X9 Only)
- Duplicate items can be detected within files or across multiple files.
 - The fields to include in the duplicate detection criteria can be selected from various fields on the check detail, return detail, and credit reconciliation record types.
 - Duplicated items can be included in reports or exported to Excel.
 - Duplicated items can be automatically deleted with trailer totals automatically adjusted.
- 25) Structurally flawed files (available for all file formats)
- X9Assist will open any and every x9 file.
 - If a file is structurally invalid, information is provided as to how many records were successfully parsed and what percentage of the overall data was analyzed prior to the failure point.
 - Records can be viewed up to the point of failure, which is extremely helpful in determining what is wrong with the file format.
- 26) Dynamic binding (available for all file formats)
- Incoming files are dynamically “bound” to configuration rules based on contents of the file header. This process allows the correct rules to be applied to the file on an automated basis without manual intervention.
 - Customer provided rules control the binding process.
 - An optional facility exists to define your upstream and downstream partners by their ABA. You can then indicate their role (upstream, downstream, both) and their status (start date, end date, etc).
 - Binding allows files to be automatically mapped to the correct validation rules.
- 27) Branding (available for all file formats)
- White label branding is a separately available product which allows our desktop tools to be branded with your corporate name, your logo, and website.
- 28) Diagnostics (available for all file formats)
- A system log is maintained which logs all events during the current user session.
 - A record trace is generated whenever a file cannot be parsed. The trace points to where the problem exists in the file and provides the technical information needed for resolution.
 - Any application aborts or exceptions are trapped and reported within the system log so they can be provided to X9Ware for research and resolution.
- 29) SDK (available for all file formats)
- Our SDK (software development kit) allows X9 files to be read, parsed, validated, and written. Your use of X9Assist as a viewer, validation tool, and data manipulation tool shows the power of this SDK.
 - The Java JRE is current required to be 1.8 or higher (released by Sun in 2006).
 - The SDK can be used as a proven base to jump-start your application development.
 - Once provided, the SDK will run without a separately provided license key as is currently being done by competitive products. Your benefit is that the SDK is key-less, which means

that you do not have to worry about it causing an unexpected time-out event at some unforeseen time in the future.

30) Technical design

- 100% Java.
- Operating system independence. The distribution is currently for Windows, but X9Assist can also be optionally run on Linux or other Java based platforms should there be a customer requirement to do so.
- X9Assist can be installed without ADMIN rights utilizing a ZIP file instead of our default Windows installer. Please contact us if you require our ZIP file for installation.
- We believe that X9Assist performance meets or exceeds other products that exist in the marketplace today. We would appreciate your feedback on this topic.

Installing X9Assist and X9Utilities

X9Assist and X9Utilities each have a variety of installation packages that are available for download from our website. All of these packages are based on X9Assist as a 100% Java application.

Minimum Requirements

X9Assist and X9Utilities have minimal hardware requirements of 4GB of RAM and 200 MB of available hard drive space. X9Assist requires an SVGA-capable video card. From a Windows perspective, both will run on a system running Windows 7 or higher. Essentially, any system that can run Windows 7 will also run X9Assist.

Java Heap size when using our Installation Packages

X9Assist and X9Utilities each create a program launcher for their runtime environment (x9assist.exe and x9util.exe, respectively). These define a JVM heap size of 4GB which can be increased through an update within the installation folder, to / app / x9assist.cfg or / app / x9util.cfg.

X9Assist Installation Packages for R4.10 and Higher

Package Type	Usage and Capabilities	ADMIN Rights Required?	Ability to Associate File Extensions?	Must a Java JVM be Pre-Installed?	64 bit or 32 bit?
EXE-BUILD is our preferred installer as long as you do not require MSI	Installs X9Assist using our EXE based installer, which can run with or without ADMIN privileges. When running without ADMIN, the software will be installed into the current USER APPDATA folder. When running with ADMIN rights, the software will be installed into the Windows Program Files folder, which is utilized for 64 bit applications.	Optional	Yes	No	64
MSI-SETUP	Installs X9Assist run-time compiled for Windows. This zip package does requires ADMIN privileges. MSI is commonly used by large organizations as part of internal software distribution system, which requires a silent-automated process.	Yes	Yes	No	64
JAR-ZIP	Installs the X9Assist JAR and associated resources only; there is no run-time JVM included in this zip. This zip package does not	No	No	Yes	Either

Package Type	Usage and Capabilities	ADMIN Rights Required?	Ability to Associate File Extensions?	Must a Java JVM be Pre-Installed?	64 bit or 32 bit?
	require ADMIN privileges.				

X9Assist Installation Packages for R4.09 and Earlier

Package Type	Usage and Capabilities	ADMIN Rights Required?	Ability to Associate File Extensions?	Must a Java JVM be Pre-Installed?	64 bit or 32 bit?
Setup-MSI	Installs X9Assist for all users on the current device. This is a Windows installer that requires ADMIN privileges. This is an MSI based installer that is zipped to facilitate distribution.	Yes	Yes	No	64
EXE-Build-Zip	Installs X9Assist run-time compiled for Windows. This zip package does not require ADMIN privileges.	No	No	No	64
JAR-Build-Zip	Installs the X9Assist JAR and associated resources only; there is no run-time JVM included in this zip. This zip package does not require ADMIN privileges.	No	No	Yes	Either

X9Utilities Installation Packages

Package Type	Usage and Capabilities	ADMIN Rights Required?	Ability to Associate File Extensions?	Must a Java JVM be Pre-Installed?	64 bit or 32 bit?
Setup-MSI	The X9Utilities for all users on the current device. This is a Windows installer that requires ADMIN privileges. This is an MSI based installer that is zipped to facilitate distribution.	Yes	Yes	No	64
JAR-Build-Zip	Installs the X9Utilities JAR and associated resources only; there is no run-time JVM included in this zip. This zip package does not require ADMIN privileges.	No	No	Yes	Either

Installation Overview

X9Assist and X9Utilities are packaged with a setup installer for Windows. Several comments:

- Each release of those products will be identified as Rx.xx.
- Within a given release, these installers will remove any previous version and will install the new version of that same release. However, older versions of these products will not be automatically removed by these installers.
- This approach allows a previous release to be retained (side-by-side) with the new release, so the capabilities of the newest release can be confirmed.
- Once a newer release has been installed and verified, an older release of that same product can then be removed.

Specific installation notes as follows:

- Admin rights may be needed to run the installer, subject to which alternative is chosen.
- X9Assist and X9Utilities are both 100% Java. The underlying byte code will execute either a 64-bit or 32-bit, subject to the JVM it that is used to run it. However, be aware that our EXE installer has a pre-packaged JVM that is 64bit; these installers can only be used on a Windows 64 bit system.
- Due to the large nature of x9.37 files, 64 bit operation is very much preferred.
- Windows support includes Windows XP through Windows 10, including Windows Waas (Windows as a Service).
- Within our technical documentation, we refer to the installation folder as the “launch folder”. For example, by default, our EXE installer will install to / Program Files / X9Ware LLC / X9Assist Rx.xx /.
- The launch folder is considered to be read only. Once installed, X9Ware will not attempt to write to any components or files within the launch folder.
- On a default basis, the installation folder will include the product name and release level.
- There is no absolute requirement that our products (X9Assist, X9Utilites, etc) be installed and run from the Program Files folder. The applications can be copied and moved to other locations of your choice within the file system. This includes the boot drive as well as secondary drives.

Installation Notes

Other points of interest are as follows:

- There are no absolute registry requirements to run any of our application software.
- A short cut can be created for x9assist.exe and loaded to your desktop. This is typically done by right clicking the EXE and then “send to desktop create shortcut”.

- X9Assist does not need to run from the program files folder; you can put the run folder anywhere you desire and then for example just add an x9assist.exe shortcut on the desktop.
- X9Assist requires a second folder which we refer to as the home folder. X9Assist requires write access to this folder. It is important to recognize that a large amount of intermediate data can be written to this folder during x9 test file creation, which means that it should be stored on a local drive (and not a network drive) for improved performance. The home folder will typically be created for the current (signed on) user as `/documents / x9_assist /`. Locating the home folder within the Documents folder guarantees user write privileges for all files within this folder.
- Our license files are encrypted text that cannot be manipulated.
- X9Assist will write the active license to file `/ Documents / x9_assist / license / elicense.txt`. If you package your license key there, then the user will not be prompted for a license key on their first run.
- X9Utilities expects the active license to be loaded into folder `/ program launch / license /`.
- You will need a procedure to update the license key each year as a new key is provided. There are several ways to do this. You can have your own installer that would distribute that one file. Or this can be done as simply as making a new elicense.txt file available through a distribution list (etc).
- Remember that there is a large amount of data written to our home folder and especially to folder `/ Documents / x9_assist / temp /`. Although it is possible to use the X9Ware properties file to relocate the x9_assist folder elsewhere (eg, to a network drive), application performance can be negatively impacted depending on the speed of that device.

Running X9Utilities in the Cloud

X9Utilities has several features that can be used to facilitate cloud based installations:

- The X9Utilities license key can be embedded in the runtime JAR (see the Licensing section for more information on how to do this).
- The X9Utilities export function can embed output images as Base64 strings in the CSV output, which eliminates the need to store the images in the file system.

Availability of New Releases

New releases of X9Assist are made available on our website at: <https://x9ware.com/downloads/>. This link will be automatically redirected to the our most current build. The new release web page will include a list of the significant changes that have been incorporated into the update.

New downloads for X9Utilities and our SDK are not made available via a public download page. Instead, we provide a download link and password to you as needed. The download will still be from www.x9ware.com, but obviously from a different web page.

New X9Utilities and SDK builds are created for each release. A download link must be requested via email (support@x9ware.com); we will then provide the link and password. You can check the X9Assist download page to determine when a new release is available. The X9Assist download page also outlines the changes that have been to these products. Because of that, the X9Assist download page is helpful to get an idea of the recent improvements made to our overall product line.

X9Assist Failed to Start

In very limited situations, X9Assist may fail to start. There can be a couple of scenarios.

One possible situation is that X9Assist is being started without a license key using an X9Assist release that is R4.03 or older. In those situations, X9Assist will issue a “buildExpired” abort, which is issued when there is no user license and the build date is two year or more older. The easiest way to resolve this issue is to just move to the most current X9Assist release. An alternative solution is to store a valid user license file into: / YOUR USER ID / Documents / x9_assist / license / elicence.txt. You should be able to obtain a valid elicence.txt file from another user within your organization.

Another situation for an X9Assist failure is that the underlying Java JVM may fail to start. This typically happens when the workstation has global Java environment variables that negatively impact our startup environment. One situation where we have seen this is with Microfocus, where the installation of those tools set global variables (_JAVA_OPTIONS and JAVA_TOOL_OPTIONS) that unfortunately will apply to all JVMs, and not just their product. With these environment variables assigned, their tools attempt to initialize in our startup environment, but will ultimately abort since the necessary Java classes are undefined.

When you change system wide environment variables within a batch script, those changes will only affect the current batch session. This means that the changes remain local to your command session itself and will not impact the use of these variables by other batch files. Because of this, batch scripts can reassign these system variables without negatively impacting other tools and processes.

Our standard X9Assist startup is based on our x9assist.exe program launcher. We offer several alternatives that can be successful in these situations:

- In the program installation folder / runtime / bin /, you will find x9assist.bat which may be successful when x9assist failed to start.
- Another alternative is to reassign Java global environment variables, which can be done using x9assistExec.bat in the program launch folder. Content of this batch script is below. We would appreciate any feedback regarding improvements.

@echo off

: Windows batch script to launch x9assist.

: This batch script can be used as the basis to create user batch scripts.

: Use setlocal to keep Environment Variable changes local to this batch file.

: The environment is restored either at end or when an endlocal is issued.

setlocal


```

: Remove all global environment variables that will impact the JVM environment.
set _JAVA_OPTIONS=
set JAVA_TOOL_OPTIONS=

: Set a local variable for the current directory.
set DIR="%~dp0"

: Launch x9assist.exe.
: Pushd is used to push the current working directory, with subsequent popd.
: Start is used to further prevent the console window from being opened.
: Maximum heap size (Xmx) is recommended as 7g for 64-bit and 1536m for 32-bit JVMs.

pushd %DIR% & start "" x9assist.exe -Xmx7g %* & popd

exit

```

X9Assist Windows Stack Trace

We sometimes must work with a Windows stack trace. No worries, send the stack trace to us and we will do the research. The below is an example, where this is an access violation (0xc0000005).

An Access Violation is a type of Exception caused when an application Reads, Writes or Executes an invalid Memory Address. If the associated address is between 0x0 and 0x10000 (64K), it most typically means that a function returned a null pointer (0x0), and the pointer was accessed without verification, resulting in the access exception.

Here is an example, where the program counter (PC) is zero, which would typically mean that a required DLL is missing.

In this case, the first entry in the stack is 00007ffe4911833. When looking at the DLL list, this address is at offset +1833 within the jimage DLL. This provides the needed information to further research the underlying exception.

```

#
# A fatal error has been detected by the Java Runtime Environment:
#
# EXCEPTION_ACCESS_VIOLATION (0xc0000005) at pc=0x0000000000000000, pid=16628, tid=17940
#
# JRE version: (15.0.4+5) (build )
# Java VM: OpenJDK 64-Bit Server VM (15.0.4+5-MTS, mixed mode, tiered, compressed oops, g1 gc, windows-amd64)
# Problematic frame:
# C  0x0000000000000000
#
# No core dump will be written. Minidumps are not enabled by default on client versions of Windows
#
#

```

```

Top of Stack: (sp=0x000000c1565fefc8)
0x000000c1565fefc8: 00007ffe4911833 000002053d60ec40

```

Dynamic libraries:

```

0x00007ff75f6e0000 - 0x00007ff75f74b000 C:\Program Files\X9Ware LLC\X9Assist R4.08\x9assist.exe
0x00007ffff0470000 - 0x00007ffff0665000 C:\Windows\SYSTEM32\ntdll.dll
0x00007ffff6800000 - 0x00007ffff673e000 C:\Windows\System32\KERNEL32.DLL
0x00007ffffd700000 - 0x00007ffffd238000 C:\Windows\System32\KERNELBASE.dll
0x00007ffffe8a0000 - 0x00007ffffe4a1000 C:\Windows\System32\USER32.dll
0x00007ffffd400000 - 0x00007ffffd62000 C:\Windows\System32\win32u.dll
0x00007ffffe870000 - 0x00007ffffe89b000 C:\Windows\System32\GDI32.dll
0x00007ffffdb00000 - 0x00007ffffdce000 C:\Windows\System32\gdi32full.dll
0x00007ffffe370000 - 0x00007ffffe40d000 C:\Windows\System32\msvcrt.dll
0x00007ffffe240000 - 0x00007ffffe340000 C:\Windows\System32\ucrtbase.dll
0x00007ffffec60000 - 0x00007ffffef39f000 C:\Windows\System32\SHELL32.dll
0x00007ffffef400000 - 0x00007ffffef430000 C:\Windows\System32\IMM32.DLL
0x00007ffffeb00000 - 0x00007ffffebad000 C:\Windows\System32\shcore.dll
0x00007ffffef4b0000 - 0x00007ffffef54e000 C:\Windows\System32\msvcrt.dll
0x00007ffffef5f0000 - 0x00007ffffef945000 C:\Windows\System32\combase.dll
0x00007ffffef4c0000 - 0x00007ffffef5ea000 C:\Windows\System32\RPCRT4.dll
0x00007ffffef2840000 - 0x00007ffffef2858000 C:\Program Files\X9Ware LLC\X9Assist R4.08\runtime\bin\jli.dll
0x00007ffffccb20000 - 0x00007ffffccb0000 C:\Windows\WinSxS\amd64_microsoft.windows.common-
controls_6595b64144ccf1df_5.82.19041.1110_none_792d1c772443f647\COMCTL32.dll
0x00007ffffef5d0000 - 0x00007ffffef67c000 C:\Windows\System32\ADVAPI32.dll
0x00007ffffefc00000 - 0x00007ffffefc9b000 C:\Windows\System32\sechost.dll
0x00007ffffea240000 - 0x00007ffffea25b000 C:\Windows\SYSTEM32\VCRUNTIME140.dll
0x00007ffffce540000 - 0x00007ffffce555000 C:\Program Files\X9Ware LLC\X9Assist R4.08\runtime\bin\vcruntime140.dll
0x00007ffffb19c0000 - 0x00007ffffb1a5b000 C:\Program Files\X9Ware LLC\X9Assist R4.08\runtime\bin\msvcrt140.dll
0x00007ffff9e1b0000 - 0x00007ffff9ed42000 C:\Program Files\X9Ware LLC\X9Assist R4.08\runtime\bin\server\jvm.dll
0x00007ffffef5c0000 - 0x00007ffffef5c8000 C:\Windows\System32\PSAPI.DLL
0x00007ffffe83d0000 - 0x00007ffffe83da000 C:\Windows\SYSTEM32\VERSION.dll
0x00007ffffd9790000 - 0x00007ffffd9799000 C:\Windows\SYSTEM32\WSOCK32.dll
0x00007ffffef550000 - 0x00007ffffef5bb000 C:\Windows\System32\WS2_32.dll
0x00007ffffd64a0000 - 0x00007ffffd64c7000 C:\Windows\SYSTEM32\WINMM.dll
0x00007ffffec420000 - 0x00007ffffec432000 C:\Windows\SYSTEM32\kernel.appcore.dll
0x00007ffffef4910000 - 0x00007ffffef491a000 C:\Program Files\X9Ware LLC\X9Assist R4.08\runtime\bin\jimage.dll
0x00007ffffec0d0000 - 0x00007ffffec2b4000 C:\Windows\SYSTEM32\DBGHELP.DLL
0x00007ffffd71c0000 - 0x00007ffffd71ec000 C:\Windows\SYSTEM32\dbgcore.DLL
0x00007ffffef50000 - 0x00007ffffef50000 C:\Windows\System32\bcryptPrimitives.dll
0x00007ffffca090000 - 0x00007ffffca0b5000 C:\Program Files\X9Ware LLC\X9Assist R4.08\runtime\bin\java.dll

```

X9Assist Installation on Linux and macOS

X9Assist (X9Validator, X9Vision, AchAssist, etc) are running successfully in these environment by several organizations with no known issues. Since the X9Assist executable is a JAR and is 100% Java, we do not anticipate any problems for this environment, but we also do not have a lot of experience with it (since virtually all of our customer installations are on Windows). These installation instructions are based on the feedback we have received from actual customer installations.

X9Assist Installation on Linux

We would appreciate your review, experience, and recommendations regarding improvements to this installation process.

1. A Java JVM must be installed if not present ... Download `jdk-19_linux-x64_bin.rpm` (typically located in the Public folder).
2. Run `sudo yum localinstall jdk-19_linux-x64_bin.rpm` to do the installation.
3. Run `sudo update-alternatives --config java` and select the version of Java just installed.
4. Download `x9assist-4.10-jar.zip` (typically located in the Public folder).
5. Unzip the file with `unzip x9assist-4.10-jar.zip` you may wish to rename the resulting `x9assistJarZip410` folder to `x9assist`.
6. Create a shell script to start `x9assist` named `startx9`. It should contain the following:

```
#!/bin/bash
cd /home/your.name/x9assist
java -Xmx7g -jar x9assist.jar
```

7. Run `chmod 754 startx9`.
8. Run the new `startx9` that was created to test the installation.
9. Registration info will need to be entered into X9Assist as part of the first launch.
10. Note that it may be necessary to run `sudo update-alternatives --config java` in order to point to the correct Java version for use with X9assist.

X9Assist Installation on macOS

We would appreciate your review, experience, and recommendations regarding improvements to this installation process. This guide will walk you through installing X9Assist, a Java Swing application, on your macOS computer.

1. A Java JVM must be installed if not present. One alternative is to use Homebrew (<https://brew.sh/>) to install OpenJDK 17 (adoptopenjdk17).
2. Download our JAR-ZIP Package for X9Assist, which is available on our website and is a complete runtime with all requirements in JAR format.
3. Unzip the downloaded X9Assist-JAR-ZIP to a folder such as `~/Applications/x9assist4xx`.
4. Create the shell script to launch X9Assist and put it in `~/Applications/x9assist5xx/x9assist.sh`. X9Ware is interested in your changes to this script, so we can make the process better for future usage. The shell script could be named `x9assist.sh` and might look like the following:

```
#!/bin/bash
SCRIPT_DIR="$( cd "$( dirname "${BASH_SOURCE[0]}" )" >/dev/null 2>&1 &&
pwd )"
pushd $SCRIPT_DIR
JAVA_HOME=$(/usr/libexec/java_home -v 11)
nohup java -jar ./x9assist.jar -> /dev/null 2>&1 < /dev/null &%
```

5. Use Automator to create a new workflow which contains a single step - "Run Shell Script". For this new workflow, use `File > Save As`, and change the File Format to "Application". This approach will open the application, run the Shell Script step, execute the command, and then exit the windows after it completes.
6. Registration info will need to be entered into X9Assist as part of the first launch.
7. You can now run and test the X9Assist application in your environment.

X9Assist Docker Containers

These are the steps we have used to install and run X9Assist within a docker container. This was done using a Windows 11 host with the container running Alpine Linux. With that background, these steps should be helpful for other similar environments.

After some research, our implementation uses Alpine Linux as its base image for several reasons. Alpine has a minimal footprint, translating to smaller image sizes, reduced storage requirements, and

ultimately, improved deployment efficiency. Alpine prioritizes security with regular updates, ensuring our containerized applications remain protected. While Alpine traditionally operates in headless mode, we can leverage Xvfb (X virtual framebuffer) within the Alpine container for graphical UI support.

Here the general steps that we used to get X9Assist running on Docker in our Windows 11 host environment. These steps will vary based on your host OS and also your specific requirements. We offer this as an example of the steps we performed, and you can hopefully use this as a jump point to get started. As with all of our documentation, your feedback would be appreciated, so we can make this as helpful for others. Steps as follows:

1. Install the 64-bit version of Docker For Windows.
2. Once Docker For Windows is installed, double check Docker For Windows properties (right click the icon on the desktop). Most importantly, it should reference terminal 0 and include the multiwindow option. It should look something like: "C:\Program Files (x86)\Xming\Xming.exe" :0 -clipboard -multiwindow
3. Install Xming for Windows. This is an X server, which is a software component that provides the infrastructure for rendering graphical user interfaces (GUIs) in the X Window System, commonly used in Unix-like operating systems such as Linux. The X server allows the graphical application running in the container to be displayed on the host's desktop environment. There are numerous choices, where we selected Xming because it is specifically designed to run X11 graphical applications on Windows systems. It is lightweight and efficient, consuming minimal system resources while providing the necessary X server functionality. We also found it to be easy to install and configure, typically requiring only a few steps to set up an X11 server on a Windows machine.
4. This X9Assist runtime is created from the JAR-ZIP package that is available for download on our website. This provides the Java JAR itself plus a series of folders that give you everything you need to get up and running. This zip file does not include a JVM. It requires that you separately install an appropriate Java JVM. As of this writing, our testing is based on Java 17, but also be advised we are keeping up with the current Java LTS releases.
5. When you download and unzip the JAR-ZIP package, you will want to add a license folder to the file structure that is copied into the container. This allows X9Assist to automatically load your product license key, thus eliminating the need to re-enter it. Look at the file structure and add a "license" folder at the same level as the other runtime folders that are present (eg, bat, files, help, images, licenseAgreement, lists, etc). Within that license folder, then add your "elicense.txt" file from your folder: / documents / x9_assist / license / . This allows the X9Assist, running within the container, to inherit and assign your license key.
6. Configure the Dockerfile, which is a text file that contains instructions needed to build a Docker image. Docker images are templates used to create Docker containers, which are lightweight, portable, and self-sufficient environments that can run applications. The docker file defines the base operating system image, required packages, file copies to be performed, environment setup, user configuration, network configuration, and startup commands.
7. Our example Dockerfile is designed for running within a Windows host (e.g., Windows 11). Because of that, it includes packages that are needed to set up an X11 server within the Linux container. This is necessary due to the fact that Windows doesn't have a native X11 server, so

the container needs these packages to display the application's GUI elements. When using Linux or macOS as the host, you can remove the installation of xorg-server, ttf-dejavu, ttf-droid, and feh packages. These packages are unnecessary, since the host's X11 server will be used. For all host environments, we must set the DISPLAY environment variable to the IP address of X11 service running on the host machine. For Windows, this is the virtual WSL IP address (Windows Subsystem for Linux) which can be obtained from “ipconfig”; just be advised that this IP address has the potential to change on every reboot. Alternatively, when using Linux or macOS, you can obtain the IP address by running “hostname -I” on your host machine.

8. The docker container runs X9Assist under a created userid of "app", which follows the principle and best practice of least privilege. Using a non-root user properly limits scope and provides access only to those files that are needed by the application. It similarly allows created files to be assigned appropriate ownership that is associated with the application itself.
9. You will often want to access your host files within the container. This can be accomplished by mounting a host folder either into your Linux container file system, or by actually replacing one of the folder references that already exist there. From a Windows host, this is an example of mapping a windows host folder as replacement for the corresponding container folder. If this is your standard starting procedure, you may want to create a batch file that contains this command and use this as your standard launch:

```
docker run -v C:\Users\X9Ware5\Documents\x9_assist\files:/home/app/x9_assist/files/
x9assist:latest
```

10. Our x9assist docker file for Windows is defined as follows. Be advised that this will need to be customized for your specific environment, especially as stated above, when you are running either on Linux or OSX.

```
# Use Alpine Linux base image with X11 support
FROM alpine:3.16 AS base

# Install necessary packages including OpenJDK 17, socat, and ping
RUN apk add --no-cache \
    openjdk17 \
    socat \
    iputils \
    xorg-server \
    ttf-dejavu \
    ttf-droid \
    feh \
    bash \
    && apk del --purge apk-tools

# Create a non-root user 'app' with UID 1000 and set its home directory
RUN adduser -D -u 1000 -h /home/app app

# Create writable directories for X11
RUN mkdir -p /tmp/.X11-unix && chown root:root /tmp/.X11-unix && chmod 1777
/tmp/.X11-unix

# Create application directory
RUN mkdir -p '/home/app/x9_assist'

# Set ownership of application directories to 'app' user
RUN chown -R app:app /home/app/
```

```
# Create a writable directory for Java's TEMP
RUN mkdir /home/app/tmp && chown -R app:app /home/app/tmp && chmod 777 /home/app/tmp

# Copy x9assistJarZip504 folder to x9_assist while preserving the folder structure
COPY --chown=app:app x9assistJarZip504/ /home/app/
RUN chmod -R 755 /home/app/

# Set working directory and ownership to user 'app'
WORKDIR /home/app/

# Assign Java temp folder which is APPDATA
ENV APPDATA=/home/app/tmp

# Set the java.io.tmpdir system property to point to the writable directory
ENV JAVA_TOOL_OPTIONS="-Djava.io.tmpdir=/home/app/tmp"

# Define a multi-stage build to avoid Xvfb in the final image
FROM base AS final

# Set the DISPLAY environment variable (replace with your actual IP)
ENV DISPLAY=172.21.80.1:0

# Remove the display 0 lock file and run the Java application
USER app
CMD ["sh", "-c", "rm -f /tmp/.X11-lock; X :0 -ac -screen 0 1280x1024x16 & java -jar x9assist.jar"]
```

11. From our experience, a more difficult part is to get X11 communications established between the docker container, the JVM, the host OS, and your X11 server. You must check that ports are not being blocked by antivirus software or firewalls. This example is using definitions that assign terminal 0 in both Xming and the docker file. We use an IP address of 172.20.80.1, since that is associated with the WSL. The Ethernet adapter vEthernet (WSL) is a virtual network interface created by the Windows Subsystem for Linux (WSL). WSL allows users to run a Linux distribution natively on Windows 10 or later. When WSL is installed and configured, it creates a virtual network adapter "vEthernet (WSL)" to facilitate network communication between the Linux environment running within WSL and the host Windows operating system. This virtual adapter has an associated IP address, allowing the WSL environment to communicate with the host machine and other devices on the network.

Ethernet adapter vEthernet (WSL):

```
Connection-specific DNS Suffix . :
Link-local IPv6 Address . . . . . : fe80::419f:3fb5:846e:9cf7%52
IPv4 Address. . . . . : 172.21.80.1
Subnet Mask . . . . . : 255.255.240.0
Default Gateway . . . . . :
```

12. Please let us know if you have questions regarding these steps.

X9Assist Automated File Copies

Several folders within the launch folder are replicated to the home folder by X9Assist. This replication is done automatically based on the two specific conditions:

- A replicated folder contains a file that does not exist in the home folder. In this situation, the file is considered as new and will be automatically copied from the launch folder to the home folder.
- A replicated folder contains a file that also exists in the home folder, but with a modification timestamp that is higher than found in the home folder. In this situation, the file is considered as updated and will be automatically copied from the launch folder to the home folder.

The following program launch folder files are automatically replicated to the home folder on each launch of X9Assist:

- / files /
- / images /
- / printFormats /
- / lists /
- / xml /

MSI Packages and Silent Operations

Msiexec.exe is the Microsoft program that interprets packages and installs products.

According to Wiki:

A package describes the installation of one or more full products and is universally identified by a GUID. A product is made up of components, grouped into features. Windows Installer does not handle dependencies between products.

A product is identified by a unique GUID (the ProductCode property) providing an authoritative identity throughout the world. The GUID, in combination with the version number (the ProductVersion property), allows for release management of the product's files and registry keys.

A package includes the package logic and other metadata that relates to how the package executes when running. For example, changing an EXE file in the product may require the ProductCode or ProductVersion to be changed for the release management. However, merely changing or adding a launch condition (with the product remaining exactly the same as the previous version) would still require the PackageCode to change for release management of the MSI file itself

Refer to this link for Microsoft documentation:

<https://docs.microsoft.com/en-us/windows/win32/msi/standard-installer-command-line-options>

Our MSI installation package is created by AIS (Advanced Installer System). Refer to this link for their package installation documentation:

<https://www.advancedinstaller.com/user-guide/msiexec.html>

The Silent Install options for the Windows installer (msiexec):

/quiet, /q, /qn	Fully silent mode
/passive	Unattended mode, shows progress bar only.
/norestart	Do not restart the system after the installation
/forcerestart	Restart the system after installation is complete
/log, /l	Enable Logging

MSI Examples

Silently install an msi package:

```
msiexec /i C:\setup.msi /qn
```

Silently install an msi package, no reboot:

```
msiexec /i C:\setup.msi /qn /norestart
```

Silently install an msi package and write the installation log to file [C:\msilog.txt](#):

```
msiexec /i C:\setup.msi /l*v C:\msilog.txt /qn
```

Silently uninstall the msi package:

```
msiexec /x C:\setup.msi /qn
```

Silently uninstall an msi package by the product code:

```
msiexec /x {xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxxx} /qn
```

Runtime Folders

Our various products utilize the following runtime folders:

Folder	Usage
/ UserID / Documents / x9_assist or x9_ware	User home folder where various files will be written by our products. This folder location is chosen since it is guaranteed to have “write” access for the current user.
/ UserID / AppData / Roaming / X9Ware LLC	X9Ware application data store.
/ UserID / AppData / Roaming / X9Ware LLC / log	System logs will be written within this folder for each program launch.
/ UserID / AppData / Roaming / X9Ware LLC / temp	Temporary (intermediate) files will be written within this folder for various functions.
/ UserID / AppData / Roaming / X9Ware LLC / options	Folder location for saved user options.
/ UserID / AppData / Roaming / X9Ware LLC / preferences	Folder location for saved user preferences.

Launching X9Assist

Location of the User Home Folder

X9Assist requires a “home folder” which is used to store various information regarding the current application session. This folder must be created in a location that can be written to by the X9Assist user. Note that this folder is not created in the installation folder (eg, Program Files), since most X9Assist users do not have ADMIN rights and hence are unable write to that folder location.

By default, X9Assist creates the home folder in your Documents folder using a folder name of “x9_assist”. This name is purposefully used to ensure it is identifiably different from our launch folder and from all other folders that are already installed.

X9Assist will store certain possibly large (intermediate) files in the user's AppData folder, since it is local to the current machine and thus provides improved performance. These files will be written to Users/currentuser/AppData/Roaming/X9Ware LLC. This default behavior can be modified via program options, using a switch setting which will move these various folders into the user AppData folder (which we refer to as the work folder).

Based on this design, the following folders can be used by X9Assist at runtime:

Home Folder (write authority)	c:/Users/UserID/Documents/x9_assist/
Work Folder (write authority)	c:/Users/UserID/AppData/Roaming/X9Ware LLC/
Launch Folder (read authority)	c:/Program Files/X9Ware LLC/X9Assist x.xx/

Some users may be required to to redirect their home folder to an alternate location based on their specific installation requirements. For example, your installation may have rules that all user files are stored on network servers, or that the Documents folder must be limited to a certain overall size, which might be insufficient to accommodate X9Assist.

You can override the location of the home folder in one of several ways:

- The easiest is to use Program Options to define a new home folder location for your X9Assist environment. This new folder must be both allocated and populated in advance, before this assignment is made. This can be easily done by first copying the existing x9_assist folder to your new location, and then by using settings on the “file locations” tab to identify the new folder location. X9Assist must be restarted to allow this setting to be established. Once you have this working, you should delete the old x9_assist folder (in your documents folder) to eliminate confusion as to where this folder is located.
- An alternative method is to specify the location of the home folder using our “properties” file. This option is a bit more complex, since the properties file will have be updated (using a text editor) which will require admin rights given that this file is stored in the program launch folder. Please reference that topic for more information regarding the content of the properties file. If you redirect the home folder location using the properties file, the new folder must be defined before you launch X9Assist and will be automatically populated on your first X9Assist

run. Your override location will not be used if the folder does not currently exist or cannot be reached.

Running From our Native Windows EXE

X9Assist is a Java application that is packaged using the standard Oracle “jpackage” tools that were introduced with Java 14. We are using jpackage to create a Windows installable package that is subsequently used as input to Advanced Installer to build a Windows setup installer. As part of packaging, we also create a ZIP file that can be used to install a runtime package on Windows along with a second ZIP package that can be used for installations on Linux and OSX.

This new distribution process is new X9Assist R4.04. We were forced into this move due to Excelsior Jet exiting the Java compiler business in early 2019. Their product was quite unique, and really our only earlier option for building a binary distributable that could be executed as a native Windows application program. Although Jet still had a JVM at its core, X9Assist was compiled for Windows and appeared in most regards as a native Windows application. This approach had its drawbacks, since the compiled code was a bit slower than that created by the Java hotspot compiler, and the garbage collection process did not perform as well. The biggest problem was that Excelsior Jet stopped updating their products as of Java8, and was seriously behind in terms of base technology and support. Bottom line is that we had to move forward when they exited the compiler business.

We now have both a short term and longer term strategy. Our short term was the R4.04 build implemented with Java 11, and subsequently R4.08 using Java 17. These use the standard Java Jpackage/Jlink capabilities to create a custom JVM, which only contains the specific modules that we require (this is a JVM subset targeted specifically for our needs). We have x9assist.exe as a launch application and have deleted java.exe and javaw.exe from the distribution, since they are not needed. We have implemented a new packaging tool (Advanced Installer) with enhanced capabilities.

In the longer term, we are going to be moving from a runtime JAR to a compiled binary, which will be done using either GraalVM or Project Leyden. Neither of these are production ready at this time. Leyden has just recently been announced and may or may not become a reality. However, they are both longer term options for us, where we would move back to a compiled binary.

Running Within a Java JVM

X9Assist can be optionally run as a Java application, which requires the installation of a JVM (Java Virtual Machine) on your workstation that is Java SE 1.8 or higher. The JVM can be either Oracle or your choice of an OpenJDK build. A 64bit JVM is definitely required given the very nature of x9.37 files with their embedded images. Linux (and other execution environments) are now fully supported by X9Ware. Contact us if you have any specific questions regarding this support.

Running Within a Java JVM on Windows

The following steps can be performed to add the X9Assist jar as an ICON on your desktop:

- Ensure that you have a Java JVM (JRE) installed per the above requirements.

- Create an icon on your desktop that will be used to launch X9Assist. You can do this in one of several ways.
 - First is to simply right click on your desktop and select new shortcut. An alternative is to browse to your Java folder, locate “javaw.exe”, right click that file, and send to desktop.
 - An alternative is to create a desktop icon which can be used as the basis for the new icon being created. This is helpful since the “javaw” reference has already been assigned. Eliminating the need to key that file location. You will then right click this icon and select properties to allow the settings to be updated..
- Set the target location to launch javaw from your installed java library along with our JAR. This parameter might look something like the following:
`"C:\Program Files\Zulu\zulu-8\bin\javaw.exe" -Xms1024m -Xmx1024m -jar "C:/Users/UserId/installFolder/bin/x9assistPro.jar" "com.x9ware.main.X9_Main" -cp x9assist_lib/`
- Set the startup folder to reference the “bin” folder within the x9assist installation folder. This will be something like the following:
`C:\Users\Users/UserId/installFolder\bin`
- Change the icon to reference “/bin/x9assist.ico”.
- Test and adjust the startup icon on your desktop as needed.

Sample JVM Batch Script for Windows

Several comments about the batch script:

- Set the current directory to the location of the jar. This assigns the startup directory and allows x9assist to inherit the correct startup folder.
- Ensure you are running a known release level of javaw. Do not just default to the currently assigned via association, since that may not be what you expect. Your default may be an EE version of Java, and we instead require SE.
- Using the start command allows the x9assist to be started and with the console box then closed. Without the use of start, the console box will remain open for the duration of your session. Also note the leading “” on the start command, which is the title for the title to display in the Command Prompt window title bar (not needed).

Batch script as follows:

```
set javaw="C:\Program Files\Zulu\zulu-8\bin\javaw.exe"
cd "C:\Users\UserId\X9Ware LLC\X9Assist R4.01\bin"
start "" %javaw% -Xms1024m -Xmx1024m -jar x9assist.jar com.x9ware.main.X9_Main -cp x9assist_lib\*
```

License Key Management

All X9Ware products are issued license keys which are directly associated to a client/company name. The client name is the assigned contact that is responsible for license usage and tracking within the company. Each license has an associated expiration date. Licenses can be perpetual, where they are assigned as expiration date of 12/31/9999. These license keys apply to all of our products:

- X9Assist (which consists of all X9+ACH desktop products).
- X9Utilities.
- X9Export.
- X9E13bOcr.
- X9Ware SDK.

X9Ware license keys are computed from the following components:

- Client/Company names
- Product type
- License expiration date

This design is focused around the following X9Ware LLC guidelines:

- Ownership of the license is tied to a specific individual who manages and tracks (as need) license key usage within your specific company.
- The license key value will vary based on the product type. This implies that you will need a new key when you upgrade from one license type to another.
- The license key is based on the expiration date. The duration is typically one year, but can be for a shorter or longer term. For example, an extended evaluation license can be issue for 60 or 90 days, and a multi-year license can be issued for companies that want to lock in pricing and simplify their license key deployment and distribution.

License Key Formats

There are two license key formats which are readily identifiable due to their different layouts:

- First is a 19-character license key that is provided through an **email transmittal**, as part of our invoicing and license key issuance process. These license keys are internally validated using a proprietary encoding scheme and will not require internet access.
- Second is a 32-character license key that is obtained from an **online purchase**. These license keys will be validated using an online registration process to our website, which will require internet access.

X9Assist License Key Registration

X9Assist license keys are entered and validated using the / Help / Registration / function which is available from the toolbar.

The two license key formats are readily identifiable due to their different layouts. You can visually inspect your specific license key and readily determine which format has been issued to you. The 19-character license keys are formatted as xxxx-xxxx-xxxx-xxxx with the embedded dashes. We have separate entry and validation procedures for these two formats.

The 19-character license key are referred to as “offline” license keys and are formatted as xxxx-xxxx-xxxx-xxxx). These keys are distributed within an email (transmittal) which is provided after your payment is either completed or confirmed as in progress. All fields (product type, customer name, company name, expiration date, and license key) are validated together as a group. After you enter the required information, you then use the register button (at the bottom of the panel) to initiate the registration process. An error in any of these fields will result in a mismatch and will thus require that you closely review and enter the information again (after the needed corrections are made). It is often easier to use copy and paste this information from your transmittal directly into the form, which will ensure that there are no typos or transcription errors.

The 32-character license keys are referred to as “online” license keys and will be registered through our website. These license keys are sent to you after an online product product from our website. As part of this registration, you will enter the product type and the license key as a functional pair, which must agree with your online purchase. After you enter the required information, you then use the register button (at the bottom of the panel) to initiate the registration process. It is often easier to copy and paste the license key into the form, which will ensure that there are no typos or transcription errors.

Online license keys will have to be manually entered as part of registration. Offline license keys do not need to be entered, because we distribute a copy of your license file that can be imported using the registration panel. If you have an off line key, then using import is much easier since it eliminates the potential for data entry errors.

If you must enter your license key information, please do so carefully. Note that you can cut and paste several of these fields (including the license key) from our transmittal which means that you do not have to re-enter this information. This can reduce the potential for input errors.

The following buttons appear on the main registration panel and are used to launch a tailored process for your product registration:

Button	Usage
Import license	Import a new license from an external file (for example, a new license that was provided to you as an email attachment). X9Ware LLC distributes these license files as part of offline purchases (not through our website). These are very small encrypted TXT files that include your specific licensing information. By importing this file, you eliminate the more tedious task of manually enter your license key. These license key files are distributed using a structured name which is formatted as: elicense_[company]_[product]_expires_[expirationDate].txt.
Enter 19-character	Enter a 19-character license key (these are formatted as xxxx-xxxx-

Button	Usage
license key	xxxx-xxxx) which would be received via an email transmittal, either from X9Ware LLC or perhaps shared with you by someone else within your organization. These keys are associated with what we refer to as an offline purchase, which means that the purchase was through a direct invoicing process (the purchase was not made online from our website). This license key will be authorized locally and will not require internet access.
Enter 32-character license key	Enter a 32-character license key that you received from an online purchase or renewal via our website. You will be able to copy and paste the key (you will not have to re-enter it). Please be aware that the license key will be validated using a real-time confirmation to our website, so internet access is required to complete this registration process.
Export license	Export your current license key to an external file that you then share with others within your organization. This exported license can be distributed as an email attachment for subsequent import.

The following button actions are available at the bottom of the online and offline popup panels:

- Register – is used to initiate registration process.
- Clear – is used as a reset function that will clear any text fields that have been entered.
- Cancel – is used to exit the registration panel.

After the license key has been successfully entered or imported, an elicense.txt file is updated within folder / documents / x9_assist / license /. This file is updated only as part of this registration process and will otherwise remain unchanged. The elicense.txt file is in an encrypted format, such that it cannot be viewed or modified. You can confirm that the elicense.txt file has been updated during the registration process using the associated date-time stamp that displayed by File Explorer.

X9Utilities / X9Export License Keys

X9Utilities and X9Export utilize the “elicense.txt” file for product authorization. The license file should be stored within the “license” folder within your program launch folder. For example, a typical installation would put the X9Utilities license file in folder / Program Files / X9Ware LLC / X9Utilities Rx.xx / license. The user license would be named as file “elicense.txt” within this folder. The license key and associated information will be logged by X9Utilities as part of each program run.

X9Ware SDK / E13B-OCR License Keys

Our SDK and E13B-OCR products are based on the “elicense.txt” file, which must be set by the application program as part of their initialization process. This approach allows the application to be self-defined and not dependent on external files, registry entries, or dongles. We feel that this design is by far the most direct way of implementing a license key for these environments, since it allows the

key to be directly embedded in the software application. In this manner, they keys work equally well for all runtime areas including development, production, and disaster recovery. This is especially true when using a perpetual key, since the key will not expire so it will never have to be touched again.

License Key Considerations

Subject to your license level, the `elicense.txt` file can be shared with others within your organization. This is done by sending it to them (as an email attachment, etc) and having them drop it into the `/documents/x9_assist/license/` folder on their respective workstation. This is then a replacement of their default or expiring license file, and will automatically bring their credentials current without having to know the license key or go through the license key entry process.

The `elicense.txt` file is unique to your company, so it should be handled accordingly and only shared within your organization, and then only within the limits of your license. For example, if you have a single seat license, you should not share your license with others, since they must obtain a unique and separate license for their workstation. You obviously need to ensure that your license is not shared outside of your organization. We have attempted to make our licensing process as painless as possible. Your cooperation and compliance is greatly appreciated.

Our licenses are issued for a duration of one or more years. You can purchase a multi-year license to eliminate expiration date issues. Our applications provide informational messages on any upcoming license expirations within sixty (60) days in advance of an upcoming expiration date.

Licensing requests can be made via email to x9assist@x9ware.com.

License Key Deployment

SDK licenses are embedded in the source code of the user developed application, making them portable across all environments. Our other products use license files that are stored in the file system and validated as runtime. In the examples below:

- “productRelease” refers to the actual product and release level that has been installed. For example, this might be “X9Assist Rx.xx” if you have installed one of our desktop products.
- “userID” refers to the current user id.

X9Assist licenses can be stored in one of two folder locations:

- Program launch folder: Program Files / X9Ware LLC / “productRelease” / license
- User home folder: Users / “userID” / documents / x9_assist / license

X9Utilites license can be stored in one of three locations:

- Program launch folder: Program Files / X9Ware LLC / “productRelease” / license
- User home folder: Users / “userID” / documents / x9_assist / license
- Embedded in the runtime JAR (for Linux, OSX, etc) as text file “elicense.txt”.

A common license lookup process attempts to locate the most applicable non-expired license for the current run-time environment. For example, X9Assist will search for a desktop license, while

X9Utilities will search for a batch license. Generally, the launch folder has precedence over the home folder.

All text files within a license folder will be examined to determine if they are a license file, and if so, their attributes. If a license file is given the name “elicense.txt”, then it is given priority over all other files in that folder location. This mechanism can provide control over the license that will be selected by our search process. Because of that, we recommend that you only store a single elicense.txt in one of these two folders (program launch or user home folders).

Storing the license file in the program launch folder can have advantages in several situations.

- When X9Assist or X9Utilities are being packaged for internal distribution, it allows the license key to be embedded within the distribution and then preassigned for users as they subsequently install the distribution package. In this situation, users will not need to be concerned about license key management. However, when using this approach, you must have a subsequent strategy to update license keys as they expire. One way to accomplish this is to create a distribution package that users can install, which contains only the updated license key and would overwrite their soon to be expired key within the program launch folder.
- When X9Assist or X9Utilities are being run from a common server (perhaps utilizing Citrix or possibly one some other common operational server), it allows the license to be stored centrally. In this situation, you can put the license key into the program launch license folder, allowing all users to automatically inherit those license key credentials.

Embedding a License in X9Utilities JAR

X9Utilities allows the “elicense.txt” file to be embedded into a runtime jar, where it will be found by our license lookup process. The jar is interrogate last, after external look-up in the program launch and user home folders have not found an appropriate license key.

Standard Java “jar” procedures can be used to store the “elicense.txt” file into the jar.

```
: Add a license key to our x9utilities jar.  
: The jar update command is easiest to use when the files references are within the current directory.  
: Jar update preserves the path name for added files, hence you want them to be unqualified.  
: This script will copy the x9utilities jar to a target diretory (in this case, our downloads folder).  
: We then copy the x9utilities license to that folder, and rename it to "elicense.txt" as part of that copy.  
: The license is then added to the x9utilities jar, the temporary copy of the license is deleted.
```

```
cd "C:\Users\X9Ware5\Downloads"
```

```
copy "C:\Users\X9Ware5\Dropbox\ProjectPackaging\release407\x9utilities407\x9utilitiesJarZip407\x9util.jar"  
x9util407WithKey.jar
```

```
copy "C:\Users\X9Ware5\Documents\x9_assist\license2\history\  
elicense_X9Ware_LLC_X9Utilities_expires_20211231.txt" elicense.txt
```

```
jar -uf x9util407WithKey.jar elicence.txt
```

```
del elicence.txt
```

```
pause
```

License History

X9Assist reorganizes licenses within folder / documents / x9_assist / license / as part of Registration Import but also as part of normal startup processing. Licenses are not moved within the program launch folder – they are only moved within the user home folder. The goal here is to organize the licenses in the following manner:

- X9Assist startup will attempt to keep the active “elicense.txt” file in the license folder.
- X9Assist startup will move all other licenses to folder / documents / x9_assist / license / history /. These may be expired licenses but may also be other licenses that are just not current active. By moving these licenses to the history folder, they are still available for / Registration / Import. When a license is moved to the history folder, it will be assigned our standard license file name based on company/client name, product name, and expiration date.
- X9Assist startup will move all remaining files that are not licenses to folder / documents / x9_assist / license / other /. These files are determined not to be licenses, so they probably did not belong in this folder in the first place.
- X9Assist function / X9Registration / Import / will comply with the above license key management strategies. If a new license is imported, the current and new licenses will be written to history (using the standardized name), and new license will also be written to / documents / x9_assist / license / elicence.txt.

MICR Fonts

X9Assist has a series of MICR E13B fonts (as shown above) that are present in our font folders and are available for image generation. All E13B fonts are encoded and subsequently printed at 8 CPI (characters per inch) as required by MICR standards.

These fonts vary only in the intensity of the individual MICR characters themselves. The provided alternate fonts may be useful in printing situations depending on your specific MICR printer, printer driver, toner, and paper quality. Please review the fonts above (note that they are not shown to scale).

Font definitions are implemented via XML definitions which can be extended to support custom MICR fonts. Please contact us if you have an alternate font that you would like to use.

Our default MICR file is ID Automation Bold. You can select an alternate MICR font within program options based on your testing and specific requirements and your MICR print results. You can print checks using these various fonts and then should select the font that is closest to the intensity of a typical bank supplied check as provided by your appropriate banking or financial institution. The selection of a font is based on your visual inspection and subsequent MICR capture testing.

MICR Font XML definitions

Each MICR font is defined by an XML file which controls the various font attributes. These definitions are distributed in the Program Files / X9Ware LLC / X9Assist Rx.xx / xml / micr folder and are then copied to your Documents / x9_assist / xml / micr folder.

The following table shows all available properties:

Property Name	Description	Format	Default Value
XmlName	The name of this MICR font definition within the xml / micr folder.	String	
FontName	The name of the associated MICR font which is stored in the fonts / micr folder.	String	
FontPointSize	Font size to be used to draw the MICR line.	Float	10.0000.
<i>FontStyle</i>	Font style to be used to draw the MICR line.	String	Plain (assigned value can be Plain or Bold).
s1 (transit symbol)	Character to be used for the MICR transit symbol which must be appropriately mapped per the defined font.	String	A

Property Name	Description	Format	Default Value
s2 (amount symbol)	Character to be used for the MICR amount symbol which must be appropriately mapped per the defined font.	String	B
s3 (OnUs symbol)	Character to be used for the MICR OnUs symbol which must be appropriately mapped per the defined font.	String	C
s4 (dash symbol)	Character to be used for the MICR dash symbol which must be appropriately mapped per the defined font.	String	D
s5	S5 symbol for CMC7 which must be appropriately mapped per the defined font.	String	E
inchesFromBottom	MICR line inches from the bottom edge of the check.	Float	0.1800
<i>inchesFromRight</i>	MICR line inches from the right edge of the check.	Float	0.3300
inchesFromLeft	MICR line inches from the left edge of the check.	Float	0.2000
inchesTransitFromRight	MICR line inches from the right edge of the check.	Float	4.3125
drawAsSingleCharacters	Indicator as to how the E13B MICR line will be formatted. Acceptable values are auto, true, and false. When enabled (values of either auto or true) then the MICR line is drawn as a single string. A value of false directs the MICR line to be drawn as single characters that are forcibly set at 8 CPI.	String	auto
antiAliasing	Indicator as to if text anti-aliasing should be used	String	auto

Property Name	Description	Format	Default Value
	when drawing the MICR line. Acceptable values are auto, true, and false.		

EOFB

EOFB (end of facsimile block) testing has created it's fair share of confusion and consternation within the Check21 image exchange community. As part of this information, we hope to provide some insight into proper EOFB recognition and validation.

Before we go deeper into this topic, we should first level set on exactly what EOFB is. From Check21 day one, the X9 image exchange standards require that all check images be encoded in TIFF format and that all image compression be done using the Group 4 Fax compression algorithm.

EOFB is a defined requirement as part of the x9.100-181 image standard which applies directly to the current x9.100-187+UCD specification, and presumably to the Canadian Standard 015 as well. EOFB does not otherwise currently apply directly to x9 image exchange.

EOFB is part of the Group 4 compression standard, and hence is mandated as part of Check21 image exchange. There is really no wiggle room in that definition. However, since TIFF images can be successfully processed without EOFB, almost all Check21 processors have historically accepted image exchange files without the presence of EOFB within the images. Because of that, EOFB has typically been more of a follow up conversion regarding image files, and not associated with hard failures and rejected files.

However, that situation is beginning to change. There has been more and more awareness of this topic within the industry. Proper inclusion of EOFB is certainly seen now as a best practice. It is becoming clear that a day may come in the future when EOFB is mandated by the major image exchange processors. It will only take several of the largest processors to go this way. If that happens, there could be a tide that flows through the industry where EOFB becomes a mandate.

Why would EOFB become an absolute mandate? There are several reasons. The first and most obvious is that it is a Group 4 Fax compression requirement, hence it is really required to be present in all TIFF images that are exchanged using the Check21 network. This position is strengthened by the firm statements within the x9.100-181 standard that EOFB is required. The second reason is that EOFB is a good indication that an image has been correctly formed and that it is entirely present. EOFB was devised as part of the Group 4 Fax standard since those images are typically transmitted by fax machines, and some indication was needed that the complete image had been both transmitted and received. The binary text within the image does not explicitly accomplish that, hence EOFB was incorporated into that standard.

What is EOFB?

When you get down to basics, each TIFF image contain one or more image segments. Each image segment is individually defined by an offset within the image (tag 273) and a data length (tag 279). Each segment contains a compressed version of 1's and 0's that represent the black and white pixels within the image itself. As an aside, tiff tag 262 (photometric interpretation) is defined as zero when the 0's represent white and as one when the 0's represent black.

Per the Group 4 Fax definition, each image segment must include an "end-of-facsimile-block" indicator at the logical end of the segment data. This is a marker that the image segment is complete and fully accounted for. Without the EOFB marker, you do not have absolute knowledge that the image segment is complete. TIFF readers many times may assume that missing rows consist entirely of white pixels. EOFB is a positive marker that image segment is present in it's entirety.

X9 Standards

Since EOFB issues can be challenging and take considerable time and effort to resolve, there is a strong argument that this validation should always be performed so conformance can be confirmed.

Per the X9.100-187 standard:

- Each G4 compressed image strip shall be terminated by a 24-bit End-of-Facsimile-Block (EOFB) code. The EOFB code is defined in the ITU-T Recommendation T.6 Specification.
- TIFF writers shall always encode the full, prescribed number of rows, with a proper EOFB immediately following in the encoding. Padding shall be by the least number of 0-bits needed for the T.6 encoding to exactly occupy a multiple of 8 bits. Only 0-bits shall be used for padding, and StripByteCount shall not extend to any bytes not containing properly formed T.6 encoding.

Note that the standard does require that the required pad bits be zero. Some encoders will correctly append the EOFB but then do not ensure that the subsequent pad bits are zero. This is incorrect per the x9.100-187 standard definition.

What does EOFB look like?

EOFB has the following definition and characteristics:

- EOFB is the three byte (24 bit) sequence x'001001'.
- Each image segment must have the EOFB sequence appended at the end.
- Each image segment is then padded with binary zeroes to fill out the last eight bit byte.
- Construction thus requires that from zero to seven pad bits be appended after the EOFB to fill out the last physical byte.
- When done correctly, the last byte within the segment data cannot be zero.

As previously described, EOFB should end with a 24 bit sequence of x'001001' that is then padded with up to 7 zero bits to fill out the last byte within the image segment. This means that the tiff image segment must end with one of the following (in hex) to be valid per Group 4 Fax specifications:

x'001001'	no bit padding required
x'002002'	logical pad of 1 bit at the end of the image segment
x'004004'	logical pad of 2 bits at the end of the image segment
x'008008'	logical pad of 3 bits at the end of the image segment
x'0010010'	logical pad of 4 bits at the end of the image segment
x'0020020'	logical pad of 5 bits at the end of the image segment
x'0040040'	logical pad of 6 bits at the end of the image segment

x'0080080' logical pad of 7 bits at the end of the image segment

The following is an example of what a TIFF image segment can look like in a hex viewer with a correctly formed EOFB. In the case provided, the EOFB marker has a value of x'004004" which is the third in the above provided list.

All of this starts to make things appear simple. Now for the complications:

- As stated earlier, there can be more than one image segment that is present within a tiff image. EOFB validation requires that every image segment be checked.
- The TIFF IFD (Image File Directory) typically appears at the front of an image with the image segment(s) appearing at the end of the image. However, that is not a requirement and many image encoders will put the IFD at the end of the image and not at the beginning. The task of visually inspecting the EOFB is much more complex when the IFD is located at the end of the image (and not at the beginning).

Validation Process

Validating the correct presence of EOFB requires inspection of the last three or four bytes of each image segment. Essentially, this inspection can be performed as follows:

- Isolate the last four bytes of image segment data.
- Continue shifting the isolated image data right, one bit at a time, until the bottom most right bit is non-zero. This process takes advantage of the requirement that the pad bits always be zero and that the 24 bit EOFB sequence ends in a non-zero bit.
- Note that no more than seven (7) shifts should be necessary.
- Validate that the resulting right most bit sequence in the rotated data is x'001001'.

How can your x9 tool provide good EOFB support?

The follows tools and capabilities really help:

1. Ability to enable or disable EOFB validation at the x9 specification level
2. Ability to flag EOFB errors at varying severity levels as you implement full validation
3. Ability to see images in hex format
4. Ability to see a parsed version of TIFF directory
5. Ability to export EOFB validation errors so they can be easily reported to the originator
6. Ability to export failed images so they can be provided to your originators
7. Ability to activate debug logging to provide more insight into failed EOF validations
8. Ability to export tracing information so you can forward to your originators

EOFB Logging

EOFB testing can be difficult. To assist in your testing and validation, it is important that your image validation tool provide feedback which allows you to easily determine image content so you can see why the image has failed EOFB testing. Without this, you believe that the image has failed EOFB but you have no proof of that and nothing to confirm your suspicion. Obviously large x9 files have a lot of

images and hence we do not want to log everything. We only want to see the images that have failed EOFB testing.

Here are logging examples. Note that logging is only performed for EOFB failures, and then only up to the logging count that is specified in program options.

2014-01-10 06:29:32.805 [INFO] EOFB validation failed for record(22); end of image segment(88888888888888888888888888888888FFFFFFFFFFFC00400) (id0)
2014-01-10 06:29:32.805 [INFO] EOFB validation failed for record(24); end of image segment(FFFFFFFFC1C1C1F111FFFFFFFFFFFFFFFFFFFFE0020) (id0)

Let's take a closer look at these two failures. Both are manufactured error situations but they do help to allow us to see actual image data and why a specific image has failed EOFB validation.

In the first example, the last four bytes of the image segment contains x'FFC00400". This segment clearly does not end in a valid EOFB bit string.

In the second example, the last four bytes of the image segment contains x'FFE00200". This segment clearly does not end in a valid EOFB bit string.

X9Assist Validation and Support

X9Assist uses TIFF rules XML to define when the EOF validation process is to be performed. Note that each defined x9 configuration can use an appropriate tiff rules XML definition. This allows you to define which x9 configurations will perform EOFB validations, and which will not.

```
<tiffControls>
  <endianFormat>littleEndian</endianFormat>
  <ifdOnWordBoundary>>false</ifdOnWordBoundary>
  <requiredAscendingTags>>true</requiredAscendingTags>
  <duplicateTagsAllowed>>false</duplicateTagsAllowed>
  <privateTagsAllowed>>true</privateTagsAllowed>
  <privateDuplicateTagsAllowed>>false</privateDuplicateTagsAllowed>
  <multiStripAllowed>>false</multiStripAllowed>
  <validateEOFB>true</validateEOFB>
  <imageMinWidth>4.100</imageMinWidth>
  <imageMaxWidth>10.500</imageMaxWidth>
  <imageMinHeight>1.752</imageMinHeight>
  <imageMaxHeight>5.700</imageMaxHeight>
</tiffControls>
```

In addition, the messages XML definition allows the severity of the EOFB error to be assigned (as either error, warn, info, ignore). Your x9 configurations can also optionally include a unique message XML definition, allowing you to automatically set the severity associated with these errors.

```
<error> <id>tiffImageSegmentDoesNotEndEOFB</id> <nbr>333</nbr> <sev>Warn</sev>
    <desc>tiff image segment does not end with EOFB</desc>
```

</error>

Finally, you can use program options to indicate that you always want to perform EOFB validations. When this option is activated, EOFB validation will be performed for all x9 files regardless of the x9 rules that are being applied. Generated EOFB error severity will then be assigned as follows:

- If the current tiff rules require EOFB validation, then the error severity is assigned from the current message rules.
- If the current tiff rules do not require validation, then the error severity is set as informational.

Summary

EOFB validation is a more complex x9 topic that is gathering a lot of interest within the x9 community. This type of validation can be done on a manual basis against one or two images. However, saying that this can be time consuming is a tremendous understatement of the effort that is required. You need automated tools to assist you in the process.

X9Assist Panel Sizes

X9Assist is designed to be as usable and intuitive as possible. This includes having functions that work as you would expect them to, and to not hide functionality behind menus or unknown shortcuts.

As with any application, much remains to be done but hopefully you will see the effort that has been put forth in this area. We recommend your suggestions to improve both functions and usability. Please send them to x9assist@x9ware.com and we will make every effort to incorporate them into the application.

Monitor size is an issue with any application, since available monitors will range from small laptop screens to very wide external monitors. X9Assist windows are designed to have as much information as possible and also to take advantage of large (wide) monitors.

X9Assist has the following features in this area:

- X9Assist has several main panels that are split windows with “sliders” that allow you to adjust the visual information based on your requirements and monitor size. An example is the x9 tree that is shown on the left side of the primary dashboard. As you adjust the size of these windows, X9Assist will save your slider positions in a preferences file and will automatically restore those settings in your next application session. This provides you with an easy way to tailor X9Assist to your usage patterns.
- X9Assist has a variety of windows (frames) through the application. Examples are generate, scrub, import, and export. These windows have initial sizes but they can also be resized by grabbing them in a corner of the window and then dragging them to your desired size. When you resize windows, your changes are saved in a preferences file and will be automatically restored in your next application session. Again, this is done to make the application as usable as possible and also to allow adjustments based on your specific monitor size.

User preferences are updated as you are using the application and are automatically saved on exit. This actual file name varies by release level and includes a suffix which is based on monitor size.

The preferences file can be deleted at any time. X9Assist will automatically revert to defaults on the next application launch, when a new preferences file will be created. Window sizes can be reset using the / Help / Reset / function which is available on the tool bar.

Error Messages

Our goal is always to make our error messages as clear as possible. Please let us know if you have issues with a specific error message and how it can be improved upon for clarity. Continuous improvement is important to us, and we need your feedback to make things better.

Error messages are complicated, because they are originated out of a validation process that is based on standards and their associated rules. Our goal has been to implement our validations and their resulting messages based on the actual standards the way that they are written, and not just based on individual perception. That said, we also know that there is a lot of “gray areas” based on how the standards are written, and situations that are not as well-defined as they should be. We also know that there are very common implementations that are widely accepted throughout the industry, where field values can be accepted even when they fall into one of the gray areas. With that as a background, our mission for file validation is to attempt to implement the standards as written, although we occasionally fall back to the “spirit” of the standard even when that differs from the actual text as written.

Message Editor

File validation process is based on a configuration, which consists of both validation rules and message definitions. If you need to customize our standard error messages, you must first create your own message xml file. These customized messages files are an extension to our standard messages and will contain only your modifications to our standard definitions. This approach simplifies the process of moving to new releases, since your messages are just overrides to our standard set.

The Message Editor allows entry of needed error message overrides. It allows error message text and severity level to be assigned based on user requirements. Changes can be made at either a global level, or at the record type and field level. The Message Editor will save your changes to a new message file which must be incorporated into your configuration set.

Error Message Patterns

Error message formatting is controlled through a pattern definition. Patterns are provided through a hierarchy (high to low) as follows:

- 1) Specific error message definition.
- 2) Global setting in a message xml extension file.
- 3) Global setting in the standard message xml file.
- 4) Properties.
- 5) Global setting by an SDK application.
- 6) System default.

The system default message pattern may change from release to release, but is generally structured as follows:

```
|{Prefix}|{Line}|field {RecordDotField}|{Name}|position {Position}|{Description};|  
value({FieldValue})|expecting({ExpectedValue})|{Comments}|failed({Reason})|[{Sev}]|
```


Raw (unstructured) text is simply copied from the pattern. Structured fields can be taken from this list:

- {Prefix} – this is typically “record” or “line” based on the file type.
- {Line} – line (record) number within the file where the error was identified.
- {RecordDotField} – record and field where the error occurred.
- {Name} – field name where the error occurred.
- {Position} – position of the error field within the record where the error occurred.
- {Description} – error description from the messages xml file.
- {FieldValue} – value assigned to the field in error.
- {ExpectedValue} – expected value based on context and calculations.
- {Comments} – possible additional information when available for this error.
- {Reason} – assigned reason code which most typically is the edit rule that failed.
- {Sev} – error severity.

Error Severities

The severity associated with a given error can be changed either globally, for that particular error message, or can be assigned for a specific record type and field. Error severity overrides are typically applied using Message Editor, with the resulting message xml file incorporated into a configuration. The following message severity levels are available:

- Error
- Warn
- Info
- Ignore

Customization

Validation of 4x4 Routings

On a default basis, our validations of routings in type 25 (check detail) and type 31 (return detail) records have been implemented exactly as they are defined by each standard. This means that we have attempted to faithfully implement these standards exactly as they have been documented.

In some organizations, there may be reasons why these standard validations are not applicable. For example, a common requirement is to accept routings in nnnn-xxxx format when that format is generated by capture systems and passed to downstream applications.

These requirements can be accommodated by making changes to our various XML rule definitions that exist in folder the program installation folder. For Windows, this would typically be either Program Files or Program Files (x86). Validation rules are defined by x9 specification within folder / program files / x9ware llc / x9assist rx.xx / rules / x9rules /. As an example, you will find the x9.37 DSTU specification defined in XML file “x9rules_dstu_2003.xml”.

Within these various XML files, you will find each x9 field defined within their respective record types. For example,

```
<field> <item>x25.04-p019-1008-mandatory-modifiable</item>
      <edit>ABA</edit>
      <name>Payor Bank Routing Number</name> </field>
```

The edit applied to a routing could then be modified to be one of these validation rules:

Validation Rule	Routings that will be accepted
<edit>ABA</edit>	Formats nnnnnnnnb (trailing blank) and nnnnnnnnC (trailing check digit validated on a MOD10 basis).
<edit>ABA4X4</edit>	Formats nnnnnnnnb, nnnnnnnnC, and nnnn-xxxx.
<edit>AbaorCpaPayorRouting</edit>	Formats nnnnnnnnb, nnnnnnnnC, nnnn-xxxx, and nnnnn-xxx.
<edit>none</edit>	All values.

Missing Image Documents

X9Assist will utilize the missing image documents in several situations:

- Generate when dynamic images are not drawn
- Import when an image is not provided and dynamic images are not drawn
- Modify when an image is forced to a missing image document

X9Assist has separate missing image documents that are located in the / x9_assist / images / folder. You can customize these images per your requirements.

These images must be stored in Tiff format and can be created using various paint tools. Our recommendation is GIMP2 which is open source and easy to use. There are obviously many other options and feel free to use your tool of choice.

Per X9 tough tiff standards, there are requirements to create a single strip Tiff image with very specific tiff tags. This is difficult (perhaps impossible) to do with available image paint tools. Given this issue, X9Assist use our internal tiff repair functionality to create a Tiff image that meets X9 tiff standards. This simplifies your process since you do not need to be concerned with Tiff standards when you create the tiff images to be used for missing front and missing back.

Credit Reconciliation Documents

X9Assist will utilize the credit reconciliation image documents when drawing the images for a type 61 credit reconciliation record.

The front image must be designed to accept the overlay of a MICR line at the bottom. The MICR line will be added by X9Assist using information from the type 61 credit reconciliation record.

These images must be stored in Tiff format and can be created using various paint tools. Our recommendation is GIMP2 which is open source and easy to use. There are obviously many other options and feel free to use your tool of choice.

As with the missing image documents, X9Assist will use your image and our repair routines to create the TIFF image that will be used internally. Because of that, you do not need to be concerned with X9 tiff standards when you create this image.

Rules and Configurations

Rules

File validation is controlled by the combination of the following rules:

	Description	Folder Location
x9rules	Contains a list of the rules that are applied to the records and fields within the file being validated. These rules are used to generate field level errors.	/Program Files/X9Ware LLC/X9Assist/rules
tiffrules	Contains a list of the rules that are applied to the tiff images within the file being validated. These rules are used to generate image related errors.	/Program Files/X9Ware LLC/X9Assist/rules
messages	Contains a list of the error messages and their associated severity levels. These rules are used to assign the severity level of the errors that are generated during the validation process. Each error has an assigned severity level of Error, Warn, Info, or OK.	/Program Files/X9Ware LLC/X9Assist/rules

Customized Rules

If you have a need to develop customized xml rule files, you should:

- Copy any such rule file to a new name that is readily identifiable.
- Make your changes to the newly created rule files.
- Reference those xml rule files in your configuration definitions.

Configurations

A configuration is a unique combination of our rule files. Each logical configuration is defined and assigned a name by using the Configuration Editor. You can build new configurations to meet your specific requirements. For example, you may have different file formats for a variety of needs:

- Exchange with your commercial customers (POD files that include debits and credits)
- Exchange with your image exchange partners

- Exchange using variant files of your specific definitions
- Internal files that are generated by your internal applications

The logical configuration name is used by the rules binder to automatically assign the rules that are used during file validation.

Configurations are loaded from config.xml during startup, where each entry defines a logical configuration that consists of x9rules, tiffrules, and messages. The content of config.xml can be maintained by the Configuration Editor, which is a desktop tool that is available in X9Validator and X9Assist.

Within config.xml, each configuration entry is defined as either system or user. System entries represent internally defined configurations that will change from release to release. These entries should not be modified. User entries can be created using the Configuration Editor to represent specific user configurations, which would be done as part of the implementation of customized validation rules.

Config.xml includes control entries that can be utilized to define several customization options:

- “loadSystemEntries” determines if the system defined entries will be loaded during application startup. This value would normally always be enabled (true) but can be disabled in those environments where a limited set of configurations is to be provided.
- “defaultX9Configuration” can be used to implement a system wide default x9 configuration which will be used as the default configuration when new files are opened and validated. X9Assist allows this system default to still be overridden at the user level through the use of program options.

These control definitions are by default as follows:

```
<configControls>
  <loadSystemEntries>true</loadSystemEntries>
  <defaultX9Configuration>Auto</defaultX9Configuration>
</configControls>
```

Backups

Our standard desktop environment will take backups of the xml rule files as a part of normal execution. These backups are written to the backup folder. Program Options can be used to control the generation of these backups and their retention. Most importantly, these backups exist and you can use them to recover information should that situation arise.

Properties File

The X9Ware properties file is used to define customized run time options that are available throughout the X9Ware application suite including the SDK. The properties file is optional with default values which work for most installations. The properties file can be allowed to default, packaged into the SDK as an internally defined resource file, or very purposefully located in the Program File folder where it will require Administrator rights to be updated.

Based on the nature of the properties file, it is usually configured and installed as part of the package for a given environment. There are situations where a modified properties file applies to all users within your organization. In that situation, it is often advantageous to define and test your custom properties file, and then include it in your own build and distribution package.

Properties File Location

The properties file is a text file defined as x9ware.properties which can be edited using your text editor of choice. This file is optional and will assign default values when not present. It must be defined on the file system within the program launch folder and typically as file /Program Files / X9Assist x.xx / x9ware.properties /. SDK users can alternatively package this file within the JAR within the properties resource folder. The location of the properties file which has been loaded will be included in the system log.

Properties File Content

The following is a sample properties file that shows the required format:

The following table shows all currently available application properties:

Property Name	Description	Format	Set From X9Assist Command Line?	Default Value
<i>workFolder</i>	Folder used to store certain system related files that are created on behalf of the current user. The work folder includes both the log and temp folders.	String	No	Default is based on your operating environment but would be typically something similar to: c:/users/CurrentUser/AppData/X9Ware LLC/.
<i>homeFolder</i>	Folder used to store files that are created on behalf of the current user.	String	No	Default is based on your operating environment but would be typically something similar to: c:/users/CurrentUser/Documents/x9_assist/.

Property Name	Description	Format	Set From X9Assist Command Line?	Default Value
<i>modificationLogFolder</i>	Folder used to store the modification log for an x9 file that is being saved.	String	No	There is no default; this value should be assigned when this feature is to be enabled. When omitted, the user can specify where the modification log folder should be saved as part of each x9 file save operation.
<i>channelBufferSize</i>	The number of bytes read for each x9 input file read request. Larger values can improve performance when files are being read from network drives.	Integer	Yes	The default value is 10485760 (which is 10MB) and has been assigned after considerable performance testing and feedback from a variety of environments. This value should be only changed when absolutely necessary.
<i>websiteExceptionReporting</i>	Website exception reporting which is used by X9Assist abort reporting (it is not used by the SDK, etc).	Boolean	Yes	True When enabled, the user will be asked if they want to open a problem report (using an input form on our website). When disabled, this will instead display a popup that informs the user of the exception.
<i>imagesAreResidentWhen64bit</i>	This flag defines whether images will be loaded and retained in memory as x9.37 files are loaded.	Boolean	Yes	true This flag is only applicable when running in 64 bit mode, and does not apply to 32 bit execution.
<i>errorMessagePattern</i>	Provides the system-wide pattern that is used for error message formatting. This is a string which defines the layout and fields that are to	String	No	[default] The default assignment string is assigned globally

Property Name	Description	Format	Set From X9Assist Command Line?	Default Value
	be included (on a default basis) within system generated validation errors. This value can also be assigned in several ways. First, through the X9Assist Message Editor, where it applies to all messages. Second, where SDK applications can insert a message pattern at either the global or message levels.			and set by the Message Manager. This layout may change from release. See the error messages topic for the layout of the error message pattern and for examples as to how this pattern string can be assigned.
<i>printListThreshold</i>	In support of Print Images, the number of page images that are held in memory before being written to a TEMP folder.	Integer	Yes	100
<i>maximumNumberOfThreads</i>	Maximum number of threads that will be used by the SDK for parallel operations. Note that this value is computed based on the number of available processors subject to this maximum limit.	Integer	Yes	4
<i>iqaMonitorLoggingInterval</i>	Interval used by the IQA Monitor to log active threads during IQA/CLM processing. This logging is helpful during problem determination.	Integer	Yes	24
<i>minimumBusinessCheckInInches</i>	Minimum size in inches for a check image to be considered as “wide” and thus be drawn using our business templates.	Float	Yes	6.500(inches)
<i>micrBandHeightPreservedInInches</i>	MICR band height that will be copied from a source image to a template image when original MICR lines	Float	Yes	0.450 (inches)

Property Name	Description	Format	Set From X9Assist Command Line?	Default Value
	are preserved by scrub.			
<i>xpsPrinterName</i>	XPS printer name to be assigned.	String	Yes	When omitted, this printer will be assigned from a system assigned default.
<i>pdfPrinterName</i>	PDF printer name to be assigned.	String	Yes	When omitted, this printer will be assigned from a system assigned default.

Properties File Sample

A sample properties file is included in the distribution which can be reviewed and modified as needed. This sample is file “properties / x9wareSampleProperties.properties”. The provided sample would then be reviewed and modified as needed based on your requirements. The modified file would then be saved in the program launch folder as described above.

Properties File Example

The following is a sample properties file that shows the required format:

```
#
# X9Ware LLC startup properties file.
#
homeFolder=g:/checkOperations/x9ware/x9_assist/
modificationLogFolder=g:/checkOperations/x9ware/fileModifications/
# end
```

User Profiles

The X9Assist User Profile function is a potential future that can be made available subject to customer interest. This documentation describes our initial thoughts on approach and implementation. This X9Assist functionality is not currently available and is envisioned as an add-on feature. Please let us know your thoughts regarding approach and applicability to your specific environment.

Our desktop tools provide a large number of powerful user tools. On a standard basis, all available functions are provided to all users subject to the current license level. For example, all X9Assist users would automatically have access to complex functions as Make, Generate, Scrub, Repair, and Modify.

In many environments, there is no need to further restrict or limit in any way the assignment of available functions (actions) to users. However, there are situations where this is not the case and the list of available user functions must be limited based on organizational or operational requirements. Here are several examples:

- Users may be located in an operational unit where they are granted update access to production files and folders in support of their overall job function but are not allowed to use tools such as X9Assist to directly update their product x9 files. In this situation, it is not possible to restrict file write access (using a security system) since that access is needed to perform core job responsibilities. You must instead limit the X9Assist modify function.
- Another variation on this is where users are located in an operational unit and are to be granted the ability to delete records but will not be granted the ability to modify records. In this case, a predefined X9Assist user profile can be assigned with the Modify function disabled. Other functions can similarly be enabled or disabled as needed.
- Finally, it may be desirable to limit IRD Print to a select group of users given the potential risk of printing duplicate items.

For these specific situations, it is possible to distribute a packaged copy of X9Assist within your environment that is customized and controlled within your organization, subject to user security levels. This is accomplished using the user profile facility.

LDAP (Active Directory) Assignments

X9Assist Unlimited has the ability to control user assigned functions subject to the groups that are assigned within your LDAP security environment. This functionality is only available to those organizations that are using X9Assist Unlimited or X9Assist Developer.

X9Assist uses LDAP as part of profile management implementation, to determine group membership for a given user. Profiles are assigned a series of available functions. Each such profile can then be mapped to one or more LDAP groups.

When a user launches X9Assist with profile management enabled, a sign-in panel is presented which requires that the user enter their user-id and password for LDAP authentication. These credentials are

used for two purposes. First to authenticate the user. Second to assign the functional profile, based on the groups which this user is assigned to.

Although groups can be existing organizational units as already defined within your directory, we recommend that you create unique new groups that map on a one-to-one basis with your X9Assist profiles. This approach will minimize group size and gives you complete control over the assignment of X9Assist functionality. Nested groups are supported for Active Directory but not for other LDAP servers. Obviously, drilling down into lower groups will add to connection time.

LDAP is highly configurable and, as a result, it is not always deployed in a standard and consistent manner across organizations. Because of this, our user authentication process is adaptable to various environments through configuration parameters. We believe that these options are sufficient and flexible enough to meet organizational needs. Please let us know as to how these parameters can be expanded and improved.

To gain access to X9Assist, users must have an existing LDAP account, and they must successfully enter the password that is associated with their account.

Profile assignment is queried on a hierarchical basis within your user profile, which is typically highest (most permissive) to lowest (least permissive).

User Permissions Xml

All LDAP parameters are defined within the “userPermissions.xml” file. Contents include:

- Definition of the profile groups that you want to use. Examples might be such groups as development, operations, support, onboarding, and so forth.
- Definition of what X9Assist functions will be extended to each profile group.
- Mapping of profile groups to your organizational units (OU) as defined within your LDAP directory server.
- Connection attributes that are needed to allow X9Assist to connect to your LDAP server, and to structure LDAP queries based on your LDAP attribute settings.
- Various other parameters such as credential retries, suspension minutes when invalid credentials are entered, etc.

The “userPermissions.xml” file must be stored in the program launch folder, which for a Windows environment would typically be in folder: / Program Files / X9Ware LLC / X9Assist R.xx / properties /. The program launch folder is used since ADMIN rights are required to update this folder, which will prevent this xml file from being modified by most users.

To utilize user permissions, your organization would typically have to package X9Assist with your permissions xml file, using your own packaging tools, and distribute this package within your environment.

Presence of the “userPermissions.xml” file automatically activates the user permissions facility.

User Permissions Skeleton

To help with configuration of your permissions xml file, a skeleton xml definition is written to the user home folder as file / documents / x9_assist / xml / defaultUserPermissions.xml /. The only purpose of this file is to provide an example of the overall content and structure of the permissions file. This file is created when each new X9Assist release is installed, since the available functions may change from release to release.

X9Assist Login Panel

When user permissions are active, the user will be presented with a login panel during X9Assist startup, where they will be required to successfully enter their user name and password. This user authentication process is done via your LDAP directory server. X9Assist only captures the user password to pass it through to your LDAP server; passwords are not stored and they are not logged.

If a user is unable to enter their user name and password, then their X9Assist session will be cancelled and the program will terminate. If they are able to enter a valid user name and password that maps to an organizational unit that is defined with the permissions xml file, then the X9Assist session will continue. Functions will be granted based on those that are enabled per the selected profile definition. Functions that are not available will be grayed out within the tool bar menus.

LDAP Parameters

The following settings can be configured:

```
Boolean debugLogging = false;
Boolean traceLogging = false;
String ldapUrl = ""; // eg, ldap://ldap.forumsys.com:389
String domainName = ""; // eg, "example.com"
String contextFactory = "com.sun.jndi.ldap.LdapCtxFactory";
String securityAuthentication = "simple";
String referralControl = "follow"; // manage referral control (RFC 3296)
String authBindDN = ""; // eg, uid=readOnlyAdminID,dc=example,dc=com
String authBindPS = ""; // eg, readOnlyAdminPassword
Boolean removeDomainSuffixWhenEntered = true;
Boolean nestedGroups = false; // true enables nested group search
Boolean activeDirectory = true; // true enables AD searches otherwise false
Boolean ssl = false; // true when ssl trust store is enabled
String sslProtocolID = "ssl";
String sslTrustStore = ""; // ssl trust store file name
String sslTrustStorePassword = ""; // ssl trust store password
String readTimeoutProperty = "com.sun.jndi.ldap.read.timeout";
Integer connectTimeout = 0; // max seconds to wait for a connection response
Integer requestTimeout = 0; // max seconds to wait for a request response
Integer searchResultsLimit = 9999; // max search results limit

Integer passwordRetries = 5; // password retries; zero for unlimited attempts
String loginLine1 = "Your security group assigns functional privileges.";
String loginLine2 = "Contact network administration if you have any questions.";

String userBaseDN = ""; // eg, dc=example,dc=com
String userIdAttribute = "uid"; // userName (DN) attribute
String userEmailAttribute = "mail"; // user email address, eg, mail or email
String userMemberAttribute = "memberOf"; // user member attribute
String userSearchFilter = "(&(objectClass=user)(sAMAccountName={0}))";
String userGroupSearchFilter = "(&(objectClass=group)(uid={0})(member={1}))";

public String groupBaseDN = ""; // eg, dc=example,dc=com
String groupNameAttribute = "cn"; // group name attribute
```

```
String groupMemberAttribute = "uniqueMember"; // group member attribute  
String groupSearchFilter = "(&(objectClass=group)(member={0}))"; // AD group filter
```

User authentication is performed by applying the user entered DN and associated password. Our user lookup is a two step process. The first lookup is done using the specified authBindDN credentials. administrator credentials. This is a directory search for the relevant user to obtain their DN. The second lookup is actual user authentication using their DN and password. This two step process also resolves the issue where individual users do not have the permissions needed for directory contextual search.

The ldapUrl parameter identifies the directory connection URL. It can start with either ldap:// (unencrypted) or ldaps:// (encrypted). We highly recommend using a secure connection. Most typically ldap:// is connected to port 389 while ldaps:// is connected to port 636.

The domainName identifies the default email domain for your users. If your directory consists only of a single domain (like example.com), then users can sign-in using just their user name (omitting @example.com), which provides a simplified and streamlined log-in. Entering the domain name at login is optional, but must be correct when entered.

The userSearchFilter and groupSearchFilter parameters are optional and used for Active Directory. These must be customized based on the specific search attributes that are needed by our AD environment. Note that the {0} string will be substituted by the current user/group name.

User Profiles Summary

User permissions is an advanced topic; as a result, you should consult with X9Ware as you begin planning this installation. Please let us if you have questions about this process or how it can be further enhanced to meet your specific needs.

Windows File Extension Association

The X9Assist installer assigns several file extensions (x9, x937, etc) to allow automated launch of our desktop application through double click from file managers such as File Explorer. Windows also has a user interface which allows this association to be performed on a manual basis.

Be advised that beginning with Windows 8, Microsoft has been implementing a new sub-system for managing file and protocol associations. Their motivation revolves around security vulnerabilities associated with allowing programs and their installers to programmatically (without user intervention or knowledge) make changes to these associations via registry updates. As we understand it, the Microsoft long term plan is to ultimately only allow the end-user to manage these associations. Because of this, the ability for programs to automatically assign these (other than from an initial application installer) has become greatly restricted.

Ultimately, these changes are coming about from a number of ways that applications work and are managed. Windows 10 and the subsequent builds are another step towards the planned target environment. However, in its current state, the file association process has become more difficult as compared to the older win32 environment that it is replacing. This has meant that the process of manually associating file extensions to applications (for direct program launch) has become more difficult. After some investigations, the easiest way to currently do this is as follows:

1. Open File Explorer
2. Locate a file with the desired file extension (for example, x9)
3. Right click that file, open with, choose another app
4. Check the box “always use this app to open these files”
5. Click more apps
6. Scroll to the bottom of the list
7. Click look for another app on this pc
8. Locate the runtime version of x9assist.exe that you want to assign
9. Select that executable and then click the open box
10. The selected file will open with the selected application program
11. This application should now be assigned at the default for this file extension
12. Do some testing to confirm your results – you can use / help/ about / to check the x9assist version that is being launched

The Windows 10 1709 update removed some of the older tools. However, on newer builds, it is still possible to directly launch these by name, even though they are not made available via the Control Panel. This commands are available either via RUN (right click the Windows icon in the lower left and then select run) or using WIN+R. Either way, then enter one of the following in the popup box:

`control /name Microsoft.DefaultPrograms /page pageFileAssoc`

`control /name Microsoft.DefaultPrograms /page pageDefaultProgram`

CSV File Requirements

X9Ware has chosen to base various processing (Import, Export, X9Utilities, etc) on the CSV (Comma Separated Value) format. We have chosen this format due to its overall simplicity and the fact that it is supported by a large number of platforms and applications. CSV reader/writer tools are readily available in most development environments. X9Ware provides CSV reader/writer classes within our Java based SDK, should customers be interested in that technical alternative.

The X9Ware CSV reader has minimum requirements, allowing it to accept the CSV files that are created by most external tools. Specifically:

- Alpha fields only need to be quoted when they contain embedded blanks.
- Numeric fields do not need to be quoted.
- Quote marks used for quoted fields can be either “ or ‘.
- Numeric fields can optionally contain leading zeroes.
- Lines can be terminated using CR (carriage return), LF (line feed), or CRLF.

Editing CSV Files

X9Ware does not recommend that you manually edit CSV files. This is only because it is so easy to create unintended issues when editing these files. These issues can be the result of user errors when these files are edited. For example leaving out a comma, inserting unintentional characters, or mistakenly removing an entire line.

If you must edit a CSV file, there are numerous tools that can be used to perform this function.

Most importantly, you cannot use Microsoft Excel to edit a CSV file, despite the fact that they like to route the CSV extension (under Windows) into Excel. The reason for this is that Excel makes horrendous and inappropriate changes to a CSV file when saved, making that output completely unusable. Excel should never be used to update a CSV file.

There are other very basic file editor options such as NotePad and NotePad++. We only recommend use of these tools for those team members that are very technically oriented and that feel comfortable using these text editors.

Other CSV Tools

We would recommend that you Google search on “csv editors” where you will be able to quickly get a list of popular tools that can be used to both view and modify CSV files. Many of these are freeware or otherwise very inexpensive. If you will have ongoing requirements to edit CSV files, then doing this review and implementing a CSV editor can help to minimize errors that will otherwise result from manual file modifications.

Editing CSV files is a complex topic. These editors typically will typically only be useful when there are a constant number of fields across all rows within the CSV file. For example, this is true when you

are creating input control file for certain utility functions (for example, x9utilities-write or x9utilities-imagePull) but will not work when there are a variable number of fields across all rows. In those situations, you will have to fall back and use a text editor such as NotePad++.

Comparing CSV Files

If there is a decision to manually edit CSV files, then you should always use a comparison utility to compare the before and after versions, to ensure that the changes are as you expect. Our comparison tool of choice would be WinMerge, which is freeware for the Windows platform and provides extensive comparison capabilities.

Summary

X9Ware realizes that there will be user requirements to edit CSV files. We want to stress that you cannot do this using Microsoft Excel. If you must edit files, we encourage you to Google search on “csv editors” and review your options. Finally, if you will be editing CSV files on a regular basis, then you should pursue a specific tool for your environment.

MICR Printers

X9Assist supports virtually all MICR printers that exist in the marketplace today. We have customers using a variety of printers and manufacturers with no issues. We have testing print drivers in the following formats:

- PCL
- PostScript
- PDF
- XPS

Our software prints images on a graphics basis, including contents of the MICR line. Since we are printing in graphics mode, there is no need for the E13B font to be downloaded to the printer, and more importantly, there is no need for a MICR specific driver. This approach has the tremendous technical benefit that we can print using generic PCL and Post Script drivers. This greatly simplifies the process and opens up a large variety of printer drivers for your consideration and implementation.

For example, suppose you want to support the Troy M402N printer. In reality, under the covers, this is actually an HP M402n laser printer. Our application fully supports the Troy M402N using the HP M402n printer driver. This printer is monochrome (blackw-white) and prints at a 1200x1200 DPI resolution. Attachment can be USB, or LAN (10/100/1000) Ethernet. From our perspective, the Troy M402N is essentially an HP M402n with MICR toner.

This also creates other printing alternatives, all of which are supported by our application. A good example is that you can use the HP M402n as your MICR printer by purchasing MICR toner which is readily available from many supplies (you can even purchase this MICR toner via Amazon).

600 vs 1200 DPI Printing

It is very important that you implement a printer driver that supports 1200 DPI. Although we can print at 600 DPI, the use of 1200 DPI is required to get the best possible MICR lines drawn on the printed page. This is a key factor to ensure that there are not digit misreads when the printed documents are subsequently scanned. If you print images and find them to be the

We do not recommend use of the the “HP Universal Print Driver for Windows PCL6” since it only supports a resolution of 600 DPI.

However, the “HP Universal Print Driver for Windows PostScript” driver does support 1200 DPI, so that is an alternative. You can also use the specific driver for your printer to achieve 1200 DPI.

Our testing has shown that most the universal print drivers from most printer vendors do support 1200 DPI printing. Using a universal PCL or PostScript driver is generally a good choice (other than the notes above regarding the HP PCL6 universal driver).

If you use a printer driver that supports a maximum of 600 DPI, then the resulting check images will not be drawn at their expected size. This problem is a result of attempting to print at 1200 DPI with scaling that is ultimately not recognized by the printer. There are two ways to resolve this problem:

- On a short term basis, uncheck the “1200 DPI print is enabled as needed” box which will reduce printing from 1200 DPI to 600 DPI. Although this will resolve the problem, it is not recommended as a long term fix, since printed images will not be at the highest quality possible. This could result in a higher digit misreads during subsequent scanning.
- On a longer term basis, we recommend that you install a printer driver that supports 1200 DPI. You can either use a specific driver provided by your printer manufacturer, or possibly use a generic driver (such as PCL or Postscript) that will work with your specific printer. You should review the technical documentation that from your printer vendor to determine your best options.

MICR Print Testing

Because of our generic approach to printing, our MICR output can be printed to virtually any printer (other than very old dot matrix printers). This means that you can do printer testing to any attached printer, thus limiting your use of the more expensive MICR toner. You can do print testing to an HP deskjet, laserjet, or any other printer device. However, we highly recommend that any such output is marked as TEST since it is clearly meant as such, and obviously will not be properly scanned by a MICR reader. A very good way of doing this is by using a yellow or red highlighter to mark the printed pages. You can come up with your best method of doing this, but best practices indicate that some proven method be performed.

Please contact us for more information should you run into printing issues.

Large Monitor / HD Screen Support

X9Assist dynamically adjusts its panel sizes to adapt to a variety of screen resolutions, which supports screen sides from 800x600 to 2560x1440 (and even larger). These tips can be used to support HD device screen resolutions (for example, 3840×2160) on devices such as the Microsoft Surface Pro.

As of R4.04, X9Assist now support high DPI resolutions through internal scaling which is provided by the application. This means that scaling does not have to be performed by Windows.

By default, this setting is set by our installer. If you have a large resolution display, there may be situations where you need to set this manually. The procedure to do so on Windows is as follows:

- 1) Right click the X9Assist icon on your desktop, for the release just installed.
- 2) Select properties.
- 3) Select the compatibility tab.
- 4) Select “Change high DPI settings”.
- 5) Check the box for “Override high DPI scaling behavior” to turn off Windows scaling.
- 6) Set “Scaling performed by” as application.
- 7) OK.
- 8) Apply.
- 9) OK.
- 10) You may have to reboot for this parameter setting to become effective.

Using a Virtual Desktop Manager

Another alternative is to use a virtual desktop manager (like Dexpot) that allows multiple desktops to be created, each of which can be assigned their own screen resolution. Using this approach, you are able to easily launch a separate desktop which is running at a reduced resolution, without to explicitly change your screen resolution. This simplifies things and eliminates the need to change the resolution back to your standard size when you are finished with your current application session.

Microsoft Surface Devices

All of the above noted alternatives can be applied to Microsoft Surface devices, which are an exceptional case given their very large screen resolutions and their use of a 3:2 aspect ratio). If you are unable to get any of the above to work, then we would suggest you adjust your screen size to a more appropriate setting:

<u>Microsoft Surface 3</u>	<u>Microsoft Surface 3 Pro</u>	<u>Microsoft Surface 4 Pro</u>
-----------------------------------	---------------------------------------	---------------------------------------

3:2 Aspect Ratio	3:2 Aspect Ratio	3:2 Aspect Ratio
Default Resolution 1920 x 1280	Default Resolution 2160 x 1440	Default Resolution 2736 x 1824
Recommended Font Scaling 100%	Recommended Font Scaling 100%	Recommended Font Scaling 100%
Recommended Resolution <u>1440 x 900</u>	Recommended Resolution <u>1440 x 900</u>	Recommended Resolution <u>1600 x 900</u>

Server and Citrix Support

There are several specific run time requirements that are required when running within either a Server or Citrix environment. These situations are somewhat unique, and specifically due to the situation where a single application instance can be used to support a large number of users, and where there can potentially be multiple concurrent user sessions that are running our desktop application.

To facility this, several command line parameters are provided to allow both the launch and home folders to be redirected from their standard locations. Although these parameters may also be set via the properties file, it may be more convenient to provide them via command line parameters as follows:

- -launch: "fully qualified path name to the program launch folder within the file system"
- -home: "fully qualified path name to the user home folder within the file system"

In addition to the above, there are several files that contain session level information that will be updated during the course of the user session based on the functions performed. These are as follows:

- User preferences are stored in folder: / userid / appdata / roaming / x9ware llc / preferences /. This information is written when the user exists the desktop application and is used to persist various data elements that describe user preferences.
- Recently opened files are stored in folder: / userid / appdata / roaming / x9ware llc / recent /. This information is written as individual files are opened and validated, and is used to support the open recent function which allows recently opened files to be quickly referenced.

Each user session will continue to be independently logged within a system log file which is allocated as a time stamped file name within folder: / userid / appdata / roaming / x9ware llc / log /. Each system log will contain information for a single user session and with no commingled data from other user sessions than may be running concurrently at the same time.

Finally, the license key will need to be installed in a common area that can be referenced by all users. The best way to accomplish this is to store the license in the program launch folder, where X9Assist has been installed (for example, C:\Program Files\X9Ware LLC\X9Assist R4.xx). The steps to install the license key are as follows:

- 1) Install X9Assist first.
- 2) Launch X9Assist and enter the license key through / Help / Registration / .
- 3) Exit.
- 4) Launch X9Assist and make sure you see "X9Assist" in the upper right corner of the title bar.
- 5) Exit.
- 6) The license key will now be in / Users / YourUserID / Documents / x9_assist / license / elicense.txt.
- 7) MOVE the license folder to the folder where X9Assist was installed.

- 8) Once the license key has been moved to this location, it will be utilized by all users.
- 9) Launch X9Assist and make sure you see “X9Assist” in the upper right corner of the title bar.
- 10) This confirms that the license is installed and operational.

CPA 015 Support

X9Assist includes extensive support for the Canadian Companion Document (CPA 015) per the Canadian Payments Association (CPA) Standard 015 specification. This functionality was initially implemented with the R2.05 release. The CPA 015 is an x9.100-187 extension and defines the x9 standard used to exchange image files between CPA member Direct Clearer Financial Institutions in Canada.

X9Assist currently has two x9 specifications that are associated with Canadian x9 support:

- `x9rules_x9.100-187_CCD.xml` is an X9Assist base x9 document which defines CPA 015 forward presentment and returned items specifications. This X9Assist definition was defined per the requirements that are outlined in the “Canadian Payments Association Standard 015” document. The X9Assist CPA 015 definition includes the addition of record type 61 (credit reconciliation) and type 68 (user record for payee endorsements). Note: this support is not specific to an industry accepted use of the Records 61 and 68 but can be used if an FI elects to use these record types in conjunction with a Remote Deposit Capture (RDC) stream or possibly cross-border exchange with a US Correspondent bank. X9Assist can be configured to an institution’s specific use of such records.
- `x9rules_x9.100-187_CCD_Print.xml` is an X9Assist x9 variant document that defines the x9 file format used by the CRD/RRD print specification as defined by Symcor. Within X9Assist, `CCD_PRINT` is defined as a variant specification that extends the base `CCD.xml`.

The X9Assist Configuration Editor supports the definition of unique x9 configurations that are used for file validation. This is standard functionality within X9Assist. Each x9 configuration consists of four components: an x9 rules file; a TIFF rules file; a messages file; and an options file. The combination of these components provides the basis for all x9 file validation within X9Assist. The most important of these is the *x9 rules file*, which defines all x9 record types, fields, and field level validations. The *TIFF rules file* is equally important and defines the validation rules that are applied to images. In a more secondary role, the *messages file* defines all error messages including their text and severity levels. Finally, the *options file* defines X9Assist program options to be applied during overall processing.

The Configuration Editor is invoked from the Editors menu bar and will display a panel that provides both inquiry and update to the configuration table.

File Format Auto Detection

When an x9 file is opened, file inspection procedures will attempt to use the appropriate x9 specification for validation based on the contents of the type 01 file control header. This automated detection is selected from the Validate function on the menu bar, using the “auto” option.

It may not be possible for X9Assist to always identify the appropriate x9 specification based on the file header content. Because of that, the Validate menu function allows the user to explicitly identify which x9 specification is to be used for the validation process. In certain situations, it may be desired to validate a file using several x9 specifications in rapid succession in order to see what types of errors are generated based on which standard is used for validation. X9Assist easily supports this type of analysis using the Validate function.

X9Assist currently has logic that attempts to identify CPA 015 and CPA 015 Print files based on the contents of the x9 file name and the type 01 file control header. This X9Assist strategy will evolve over time and X9Ware LLC certainly welcomes user comments as to how we can improve the process.

The CPA 015 auto-detect process utilizes several mechanisms, which are as follows:

- CPA 015 files are identified using the standard file naming conventions that are documented within the CPA Standard 015 specification. An incoming x9 file name that matches this detection process will be internally assigned the UCD Indicator of “C” to facilitate mapping by the X9Assist binder. This detection checks the x9 file name to contain:
 - Exactly thirty three (33) characters excluding the extension.
 - A four digit numeric Delivering Direct Clearer in positions 1 – 4.
 - A four digit numeric Receiving Direct Clearer in position 5 – 8.
 - A three character region identifier in positions 9 – 11.
 - A four character collection type (either ICP1 or ICP3) in positions 12 – 15.
- CPA 015 files are also identified based on the contents of the destination and origin routing numbers in the type 01 file header. This assignment first requires that the standards level be set to “30”. The binder will then determine if the immediate destination and immediate origin RT fields are both in ‘CP00RSNNN’ format, then the x9 file is assumed to be CPA 015. Similarly, if the immediate destination RT is in the ‘CP00RSNNN’ format and the immediate origin RT is in ABAm010 format, then the x9 file is assumed to be CPA 015. In this situation, X9Assist will internally assign the UCD Indicator to a value of “C” to facilitate mapping by the X9Assist binder.
- CPA 015 PRINT files are identified using the standard file naming conventions that are documented within the Symcor CRD / RRD print specification. An incoming x9 file name that matches this detection process will be internally assigned the UCD Indicator of “P” to facilitate mapping by the X9Assist binder. This detection checks the x9 file name to contain:
 - At least twenty five (25) characters, excluding the extension;
 - A currency indicator of 0 or 1 in position 1.
 - A file type value of 1 or 3 in position 2.
 - Reserved digits of “00” in positions 3-4.
 - A numeric two character region in positions 5 – 6.
 - A numeric three character direct clearer number in positions 7 – 9.
- X9 files not matching the above criteria are assumed to NOT be a CPA 015 file. X9Assist will continue with the automated x9 file type detection process, which will result in the assignment of one of the alternate x9 specifications (UCD, x9.100-187, etc).

Mapping

The auto-detect validation process within X9Assist utilizes a mapping table to determine which x9 configuration will be used to initially validate a given x9 file that is opened. The X9Assist Mapping Editor is used to view and modify the mapping definition.

Automated mapping is normally accomplished using fields that are readily available within the Type 01 file header record. However, in the case of the CPA 015, there are no fields that readily identify that this file was created using the CPA 015 specification. When the file was created using the CPA 015, X9Assist cannot directly determine a CPA 015 image exchange versus CPA 015 Print file simply by examining the file header fields.

During automated detection, X9Assist responds to this issue with the internal assignment of “C” and “P”. The assignment of these values to the UCD Indicator can then be used by X9Assist mapping.

The X9Assist mapping process can be used in either simplistic or very complex ways. In the most simplistic usage, the mapping of an incoming x9 file to an X9Assist x9 configuration is based on the contents of standards level and the UCD Indicator. For the CPA 015, this process then takes advantage of the internally assigned “C” and “P” values that were assigned during detection.

X9Assist automated detection of CPA 015 files can be disabled by removing the corresponding entries from the mapping table.

CPA 015 Region Table

The CPA 015 region table is defined within the ProgramFiles/X9WareLLC/X9Assistxxx/rules/tables folder as file ccdRegion.xml. This table is important to the detection process, since it defines valid regions per the ‘CP00RSNNN’ format used for immediate destination and immediate origin RTs in the file header. This table is defined in XML format and can be both viewed and modified using a standard editor such as NotePad.

```
<?xml version="1.0" encoding="UTF-8"?>
<x9table>
  <entry> <key>0</key> <value>Vancouver</value> </entry>
  <entry> <key>1</key> <value>Montreal</value> </entry>
  <entry> <key>2</key> <value>Toronto</value> </entry>
  <entry> <key>3</key> <value>Halifax</value> </entry>
  <entry> <key>7</key> <value>Winnipeg</value> </entry>
  <entry> <key>8</key> <value>National</value> </entry>
  <entry> <key>9</key> <value>Calgary</value> </entry>
</x9table>
```

CPA 015 Direct Clearer Financial Institution Table

The CPA 015 financial institution table is defined within the ProgramFiles /X9WareLLC/X9Assistxxx / rules / tables / folder as file ccdFI.xml. This table is important to the detection process, since it defines valid Direct Clearer financial institutions per the 'CP00RSNNN' format used for immediate destination and immediate origin RTs in the file header. This table is defined in XML format and can be both viewed and modified using a standard editor such as NotePad.

```
<?xml version="1.0" encoding="UTF-8"?>
<x9table>
  <entry> <key>001</key> <value>BMO</value> </entry>
  <entry> <key>002</key> <value>BNS</value> </entry>
  <entry> <key>003</key> <value>RBC</value> </entry>
  <entry> <key>004</key> <value>TD</value> </entry>
  <entry> <key>006</key> <value>NBC</value> </entry>
  <entry> <key>010</key> <value>CIBC</value> </entry>
  <entry> <key>016</key> <value>HSBC</value> </entry>
  <entry> <key>039</key> <value>LBC</value> </entry>
  <entry> <key>177</key> <value>BOC</value> </entry>
  <entry> <key>219</key> <value>ATB</value> </entry>
  <entry> <key>815</key> <value>FCDQ</value> </entry>
  <entry> <key>869</key> <value>Central1</value> </entry>
</x9table>
```

CPA 015 Validations

The following is a list of the “new” field level validations that were implemented within X9Assist in support of the CPA 015. These edits are invoked from the various x9 rules documents at the field level.

CcdFileRoutingFormat, // per CPA: currency, payment type, region, site, and FI Number
CcdOrigRoutingFormat, // validate the origin RT against the destination RT
CcdOrigRoutingFormatWhenCcd, // validate origin against destination when CCD format
CcdFileRoutingFormatOrABAmo10, // either CcdFileRoutingFormat or ABAmo10 format
ABAccd, // aba in Canadian format (nnnnnn-nnn, nnnnnnnnnc, nnnn-nnnn)
Date20yymmdd, // 2000-01-01 >= Date <= 2099-12-31
CompareToFileHeaderCollectionType, // compare collection type CCD file header RT
CompareToCommonBusinessDate, // common business date across all cash letters
CashLetterIdsUnique, // cash letter id is unique within file
BundleSequenceIsAscending, // bundle sequence is ascending within cash letter
MaximumAmountWhenCanadianCurrency, // maximum amount when Canadian currency
CashLetterDocumentationTypeVsRecordType, // value C when 10.8 = E and G when 10.8 = I

CPA 015 Summary

X9Assist has implemented a very robust validation process for the CPA 015 specifications. X9Assist has implemented every field level edit and validation as outlined within the CPA Standard 015 or

Symcor CRD Print Specification. X9Ware LLC is always very interested in user comments regarding our products and specifically how we can improve them. Please inform us of any discrepancies you identify during your use of X9Assist for CPA 015 validations and we will respond to those issues immediately.

X9.37 Type 24-34 Image Archive Record Support

The type 27 and 34 image archive records are identical in format other than record type. Each has a fixed format (with the variable size indicator set to “0”) and a variable format (with the variable size indicator set to “1”). X9Assist supports both the fixed and variable formats.

Type 27-34 Fixed Format Record

The fixed length records are defined as our format “000”. The following layout applies to the type 27 with the type 34 the same other than record type.

FIELD	FIELD NAME	USAGE	POSITION	SIZE	TYPE
1	Record Type	M	01 – 02	2	N
2	Variable Size Record Indicator	M	03 – 03	1	N
3	Microfilm Archive Sequence Number	C	04 – 18	15	NB
4	Length of Image Archive Sequence Number	M	19 – 22	4	N
5	Image Archive Sequence Number	M	23 – 56	34	NB
6	Description	C	57 – 71	15	ANS
7	User Field	C	72-75	4	ANS
8	Reserved	C	76-80	5	B

Type 27-34 Variable Format Record

The variable length records are defined as our format “001”. The following layout applies to the type 27 with the type 34 the same other than record type.

FIELD	FIELD NAME	USAGE	POSITION	SIZE	TYPE
1	Record Type	M	01 – 02	2	N
2	Variable Size Record Indicator	M	03 – 03	1	N
3	Microfilm Archive Sequence Number	C	04 – 18	15	NB
4	Length of Image Archive Locator	M	19 – 22	4	N

5	Image Archive Locator	M	(23 – (22+X)	Variable (X)	ANS
6	Description	C	(23+X) – (37+X)	15	ANS
7	User Field	C	(38+X) – (41+X)	4	ANS
8	Reserved	C	42+X) – (46+X)	5	B

Make Support

Make contains support to create both the fixed and variable length formats of the type 27 and type 34 image archive addendum records.

You should note the specific formats of these record types. The fixed length version has the image archive sequence number (field 5) defined as “numeric blank” with a fixed length of 34. The value assigned to this field must contain numeric data only.

Your use case file should have an assigned column that contains the image archive locator (field 5) defined as “ANS” and variable length. As a result, you can assign alphanumeric data to this field per your formatting requirements, and with your required length.

You must decide if you want to create the fixed or variable length version of the type 27/34 records. This is done based on the data that is provided to Make, as follows:

“data”	Fixed length 27/34 format	Your data must be numeric and can have a length of up to 34 bytes.
“length data”	Variable length 27/34 format	You data can be alphanumeric and can be formatted per you requirements and length. Note that these values are separated with the pipe (“ ”) character. The provided data will be left justified and padded with blanks on the right per your specified length. You just ensure that the provided data does not exceed your defined length.

X9.37 Type 61-62 Credit Record Support

The format of a credit record has been a complex issue since the inception of the x9 standard, since the standard only covered the format of debits (and not credits). Up until the release of x9.100-187-2013 (which contains the type 62 credit reconciliation record), there was a gap within the industry as to the formats of a x9 customer credit record. This gap was filled with several generally accepted credit formats. These formats are also typically modified by many institutions and third party processors to meet their specific needs.

Impact on Trailer Records

Credits do not typically add to the item count and total amount fields that are present in the bundle trailer (type 70), cash letter trailer (type 90), and file control trailer (type 99) records. On a standard basis, credits will only be included in the type 99 record count.

Obviously, adding credits into the total amount essentially will double count the credit in the overall amount totals, since there will also be offsetting debits that represent this amount. The industry approach is to thus not add to total amount. Once you do not add to total amount, it similarly does not seem appropriate to add to the item count. This does not create a structure issue, since the credit records will still be represented in the total record count in the file control trailer record.

Some vendors require that credits be added into the type 70, 90, and 99 records. X9Assist can support this through an x9 rules definition with additional XML parameters at the x9 controls level.

The following XML parameter will allow credits to be added to the item count in the type 70, 90, and 99 records:

```
<creditsAddToItemCount>true</creditsAddToItemCount>
```

The following XML parameter will allow credits to be added to the total amount in the type 70, 90, and 99 records:

```
<creditsAddToTotalAmount>true</creditsAddToTotalAmount>
```

Type 61 Credit Support

X9Assist has flexible support structure for type 61 credit reconciliation records. This support allows you to take one of three directions with your support for the type 61 record:

Out of box, X9Assist supports the “Metavante” and “DSTU” credit formats, which are the generally accepted credit formats that appear within the financial industry. These credit formats are defined in our standard XML rule configurations.

When using this strategy, X9Assist will evaluate each type 61 credit record and will determine the format to be either Metavante or DSTU. This evaluation is done on an automated basis. X9Assist will then use the appropriate type 61 format to map the incoming credit record.

X9Assist will also validate that all credits within the same x9 file are defined using the same type 61 format, and will throw a validation error if multiple type 61 formats are used.

As part of this strategy, X9Assist has the following credit formats defined and distributed within our XML rules on a standard basis:

Format	Record Length	Description
61-000	80	A generic credit format that contains only the two digit record type (61) followed by a 78 byte reserved field. This credit format is used when X9Assist is unable to determine the format of the type 61 record using its automated inspection process.
61-001	80	The “Metavante”™ type 61 credit format.
61-002	80	The “DSTU” type 61 record format.
61-003	84	The “X9.100-180” credit format. Although the x9.100-180 specification was generally not adopted within the industry, the credit format has been utilized by various organizations.
62-000	100	The “X9.100-187-2013” type 62 credit format.

You can alternatively disable our automated support that will accept the Metavante and DSTU formats and instead accept one or the other. You accomplish this by building a new x9 XML definition that includes 61-000 and then either the 61-001 or the 61-002 credit definitions. By having one of these (but not both), you can explicitly indicate which of the credit formats that shall be accepted. During initialization, X9Assist looks at the x9 XML rules and determines which credit formats are present and thus are to be identified and accepted. When using this approach, you should still define the 61-000 format since it will be assigned as a default by the automated identification process.

Finally, you can instead implement your own type 61 credit format as a custom format. When you do this, you should create your own x9 XML definition where you remove 61-001 and 61-002 and instead add a new 61-010 (your custom type 61 credit reconciliation format). X9Assist will reference fields within your 61-010 format on a field name basis, hence it is critical that the standard credit names be assigned to your fields. These standard names are as follows:

RECORD_USAGE_INDICATOR	"Record Usage Indicator"
AUXILIARY_ONUS	"Auxiliary Onus"
EXTERNAL_PROCESSING_CODE	"External Processing Code"
PAYOR_BANK_ROUTING_NUMBER	"Payor Bank Routing Number"
MICR_ONUS	"MICR On-Us"
AMOUNT	"Amount"
ECE_ITEM_SEQUENCE_NUMBER	"ECE Institution Item Sequence Number"
DOCUMENTATION_TYPE_INDICATOR	"Documentation Type Indicator"
TYPE_OF_ACCOUNT	"Type of Account"

SOURCE_OF_WORK

"Source of Work"

WORK_TYPE

"Work Type"

DEBIT_CREDIT_INDICATOR

"Debit Credit Indicator"

Credit Format 61-001 (Metavante™)

FIELD	FIELD NAME	USAGE	POS	SIZE	TYPE
1	Record type	M	01 – 02	2	N
2	Auxiliary On-Us	C	03 – 17	15	NBSM+D
3	External Processing Code	C	18	1	ANS
4	Payor Bank Routing Number	M	19 – 27	9	N
5	Credit Account Number - On-Us	C	28 – 47	20	NBSM-OS
6	Item Amount	M	48 – 57	10	N
7	ECE Institution Item Sequence Number	M	58 – 72	15	NB
8	Documentation Type Indicator	C	73	1	AN
9	Type of account code	C	74	1	AN
10	Source of work code	C	75	1	AN
11	Work Type	C	76	1	AN
12	Debit Credit Indicator	C	77	1	AN
13	Reserved	M	78 – 80	3	B

Credit Format 61-002 (DSTU)

FIELD	FIELD NAME	USAGE	POS	SIZE	TYPE
1	Record type	M	01 – 02	2	N
2	Record Usage Indicator	M	03	1	AN
3	Auxiliary On-Us	C	04 – 18	15	NBSM+D
4	External Processing Code	C	19	1	ANS
5	Payor Bank Routing Number	M	20 – 28	9	N
6	Credit Account Number - On-Us	C	29 – 48	20	NBSM-OS
7	Item Amount	M	49 – 58	10	N
8	ECE Institution Item Sequence Number	M	59 – 73	15	NB
9	Documentation Type Indicator	C	74	1	AN
10	Type of account code	C	75	1	AN
11	Source of work code	C	76 –	2	N

			77		
12	Reserved	M	78 – 80	3	B

Credit Format 61-003 (X9.100-180)

FIELD	FIELD NAME	USAGE	POS	SIZE	TYPE
1	Record type	M	01 – 02	2	N
2	Record Usage Indicator	C	03	1	AN
3	Auxiliary On-Us	C	04 – 18	15	NBSM+D
4	External Processing Code	C	19	1	ANS
5	Posting Bank Routing Number	M	20 – 28	9	N
6	Posting Account Number (On-Us)	C	29 – 48	20	NBSM-OS
7	Item Amount	M	49 – 62	14	N
8	ECE Institution Item Sequence Number	M	63 – 77	15	NB
9	Documentation Type Indicator	C	78	1	AN
10	Type of account code	C	79	1	AN
11	Source of work code	C	80 - 81	2	N
12	Reserved	M	82 - 84	3	B

Type 62 Credit Support

The type 62 credit record was introduced with x9.100-187-2013 and is intended to provide a standard credit record that can adopted and used throughout the industry. The type 62 record is a bit unique in that it is a fixed length record with 100 characters (instead of the more standard length of 80).

Along with the type 62 credit record definition, the x9.100-187-2013 specification includes new fields in the bundle trailer (70), cash letter trailer (90), and file control trailer (99) that indicate if the type 62 credit record (item count, image count, and amount) are included in those respective trailer record counts and amounts. Those fields should be appropriately populated when building x9 files based on this standard and allow subsequent x9 file validations to take the creator's directive into consideration.

The type 62 credit record is generally allowed to occur anywhere within the x9 file (after the type 01 file header record and before the type 99 file trailer). It may or may not have associated images.

An x9 file may have one or more type 62 credit records. A file will typically comprise one or more logical deposits, which each consist of credit(s) offset by debit(s). There is no defined standard as to whether credits would appear first or last within the deposit.

Format of the credit record is as follows:

FIELD	FIELD NAME	USAGE	POS	SIZE	TYPE
1	Record Type	M	01 – 02	2	N
2	Auxiliary On-Us	C	03 – 17	15	NBSM
3	External Processing Code	C	18 – 18	1	NS
4	Posting Bank Routing Number	M	19 – 27	9	N
5	On-Us	C	28 – 47	20	NBSMOS
6	Item Amount	M	48 – 61	14	N
7	Credit Item Sequence Number	CM	62 – 76	15	NB
8	Documentation Type Indicator	C	77 – 77	1	AN
9	Type of Account Code	C	78 – 78	1	AN
10	Source of Work Code	C	79 – 80	2	N
11	User Field	C	81 – 96	16	ANS
12	Reserved	M	97 – 100	4	B

X9.37 Type 68 User Record Support

The type 68 record is a “free form” record that was provided within the x9 specification, with the express intent to be adaptable to user specific purposes. The type 68 record is variable length and specifically structured to allow the content to be user defined.

The type 68 user record is defined as follows:

FIELD	FIELD NAME	USAGE	POSITION	SIZE	TYPE
1	Record Type	M	01 – 02	2	N
2	Owner Identifier Indicator	M	03 – 03	1	AN
3	Owner Identifier	C	04 – 12	9	ANS
4	Owner Identifier Modifier	C	13 – 32	20	ANS
5	User Record Format Type	M	33 – 35	3	AN
6	Format Type Version Level	M	36 – 38	3	N
7	Length of User Data	M	39 – 45	7	N
8	User Data Note: X = value in Length of User Data (Field 7)	M	46 – (45+X)	Variable (X)	User Discretion

The conceptual design of the type 68 record allows entities to define their own unique type 68 formats, to meet their specific x9 requirements. The entities are self identified using fields 2 (Owner Identifier Indicator), field 3(Owner Identifier), and field 4(Owner Identifier Modifier). The entities can then provide their user community with information regarding content, purpose, and interchange.

Within an x9 file, the occurrence of a given type 68 record format is determined by field 5(User Record Format Type) and field 6(Format Type Version Level). Since these record formats are by their very nature variable in length, field 7(Length of User Data) is provided to fully define the length of the user data, which is needed for both parsing and validation.

Field 8 (User Data) is a placeholder for the specific fields that are defined for this type 68 record. Each type 68 user record will have its own layout that consists of one or more consecutive fields that make up this user data area. Within this space, each such field would be assigned their attributes including usage, position, size, type, and validation requirements.

Values for field 2 (Owner Identifier Indicator) are as follows:

Defined Values:

‘0’ (not used)

'1'	Routing Number
'2'	DUNS Number
'3'	Federal Tax Identification Number
'4'	X9 Assignment
'5'	Other

X9Assist Support for the Type 68 Record

X9Assist supports the type 68 record in one of two manners:

- On a generic basis, X9Assist provides type 68 record support using the eight field content and variable length design, per x9 standards. This is done using type 68 record format “000”, which outlines the generic record and is defined within all of our x9 XML rules. These default rules can be modified by the user based on their specific needs. For example, you can use our XML rules to define the location (within the x9 file) of the type 68 records that you want generically accepted. Or, you could alternatively remove the type 68 record definitions from our XML rules, which would allow X9Assist to generate errors if any type 68 records are received within an x9 file.
- On a specific basis, where users can add their own type 68 record formats to the XML rule definitions. Doing so provides several distinct advantages. For a given type 68 format, you can define each field along with the attributes (field name, position, type, values, etc). This will allow validation to be applied to each of your fields. Another advantage is that the X9Assist field viewer will then be able to display the content of your type 68 user records on a logical field level basis. If you would like to define your own type 68 user record formats, we recommend that you contact X9Ware and allow us to provide guidance based on your specific requirements.

Defining Your Type 68 User Formats in X9 XML Rules

X9Assist allows multiple formats to be defined for a given x9 record type. Each of these formats is defined using a three (3) character format string. Format “000” is defined as the default format for each x9 record type.

Record formats are defined as part of the X9 XML rule support within X9Assist, which must be defined for each of the x9 formats that have been configured within the X9Assist runtime environment. These x9 rules are defined in folder ProgramFiles/X9WareLLC/X9Assist.../rules/ x9rules. .

Defining your type 68 records to X9Assist provides several very important advantages:

- 1) X9Assist will apply validation rules to each field within your type 68 user record.
- 2) X9Assist will display your individual fields by name in the x9 field viewer.
- 3) Make can be used to create your type 68 user record using data provided via your reformatter.

The type 68 format “000” is defined as follows:

```

<x9record>
  <type>68</type>
  <format>0</format>
  <name>User Record</name>
  <length>v45</length>
  <location>=any</location>
  <field> <item>x68.01-p001-l002-mandatory-notModifiable</item>
    <edit>n</edit>
    <name>Record Type</name> </field>
  <field> <item>x68.02-p003-l001-mandatory-modifiable</item>
    <edit>an</edit>
    <values>=0|1|2|3|4|5</values>
    <name>Owner Identifier Indicator</name> </field>
  <field> <item>x68.03-p004-l009-mandatory-modifiable</item>
    <edit>ans</edit>
    <name>Owner Identifier</name> </field>
  <field> <item>x68.04-p013-l020-conditional-modifiable</item>
    <edit>ans</edit>
    <name>Owner Identifier Modifier</name> </field>
  <field> <item>x68.05-p033-l003-mandatory-modifiable</item>
    <edit>an</edit>
    <name>User Record Format Type</name> </field>
  <field> <item>x68.06-p036-l003-mandatory-modifiable</item>
    <edit>n</edit>
    <name>Format Type Version Level</name> </field>
  <field> <item>x68.07-p039-l007-mandatory-modifiable</item>
    <edit>n</edit>
    <variableLengthDescriptor>true</variableLengthDescriptor>
    <name>Length of User Data</name> </field>
  <field> <item>x68.08-p046-l000-conditional-modifiable</item>
    <edit>ans</edit>
    <validate>>false</validate>
    <name>User Data</name> </field>
</x9record>

```

The payee endorsement record is another example of a type 68 user record, which can be reviewed for further understanding of this process:

```

<x9record>
  <type>68</type>
  <format>1</format>

```

```

<name>Payee Endorsement Record</name>
<length>f335</length>
<location>=a25|a25g</location>
  <field> <item>x68.01-p001-l002-mandatory-notModifiable</item>
    <edit>n</edit>
    <name>Record Type</name> </field>
  <field> <item>x68.02-p003-l001-mandatory-modifiable</item>
    <edit>an</edit>
    <values>=4</values>
    <name>Owner Identifier Indicator</name> </field>
  <field> <item>x68.03-p004-l009-mandatory-modifiable</item>
    <edit>ans</edit>
    <values>=000000001</values>
    <name>Owner Identifier</name> </field>
  <field> <item>x68.04-p013-l020-conditional-modifiable</item>
    <edit>ans</edit>
    <name>Owner Identifier Modifier</name> </field>
  <field> <item>x68.05-p033-l003-mandatory-modifiable</item>
    <edit>an</edit>
    <values>=001</values>
    <name>User Record Format Type</name> </field>
  <field> <item>x68.06-p036-l003-mandatory-modifiable</item>
    <edit>n</edit>
    <values>=1</values>
    <name>Format Type Version Level</name> </field>
  <field> <item>x68.07-p039-l007-mandatory-modifiable</item>
    <edit>n</edit>
    <values>=290</values>
    <name>Length of User Data</name> </field>
  <field> <item>x68.08-p046-l050-mandatory-modifiable</item>
    <edit>ans</edit>
    <name>Name of Payee</name> </field>
  <field> <item>x68.09-p096-l008-conditional-modifiable</item>
    <edit>Dateyyyymmdd</edit>
    <name>Endorsement Date</name> </field>
  <field> <item>x68.10-p104-l009-conditional-modifiable</item>
    <edit>n</edit>
    <edit>ABAMod10</edit>
    <name>Bank Routing Number</name> </field>
  <field> <item>x68.11-p113-l020-conditional-modifiable</item>
    <edit>ans</edit>
    <name>Bank Account Number</name> </field>
  <field> <item>x68.12-p133-l020-conditional-modifiable</item>
    <edit>ans</edit>

```

```

    <name>Customer Identifier</name> </field>
  <field> <item>x68.13-p153-l050-conditional-modifiable</item>
    <edit>ans</edit>
    <name>Customer Contact Information</name> </field>
  <field> <item>x68.14-p203-l008-conditional-modifiable</item>
    <edit>ans</edit>
    <name>Store/Merchant/Processing Site Number</name> </field>
  <field> <item>x68.15-p211-l025-conditional-modifiable</item>
    <edit>ans</edit>
    <name>Internal Control/Sequence Number</name> </field>
  <field> <item>x68.16-p236-l004-mandatory-modifiable</item>
    <edit>Timehhmm</edit>
    <name>Time</name> </field>
  <field> <item>x68.17-p240-l030-conditional-modifiable</item>
    <edit>ans</edit>
    <name>Operator Name/Initials</name> </field>
  <field> <item>x68.18-p270-l005-conditional-modifiable</item>
    <edit>ans</edit>
    <name>Operator Number</name> </field>
  <field> <item>x68.19-p275-l030-conditional-modifiable</item>
    <edit>ans</edit>
    <name>Manager/Supervisor Name/Initials</name> </field>
  <field> <item>x68.20-p305-l005-conditional-modifiable</item>
    <edit>ans</edit>
    <name>Manager/Supervisor Number</name> </field>
  <field> <item>x68.21-p310-l015-conditional-modifiable</item>
    <edit>ans</edit>
    <name>Equipment Number</name> </field>
  <field> <item>x68.22-p325-l001-conditional-modifiable</item>
    <edit>an</edit>
    <name>Endorsement Indicator</name> </field>
  <field> <item>x68.23-p326-l010-conditional-modifiable</item>
    <edit>ans</edit>
    <name>User Field</name> </field>
</x9record>

```

Creating an x9 XML Rules Specification for Your Type 68 User Record

X9Assist allows x9 rule specifications to be defined as either basis documents (which are independent definitions) and overrides documents (which are changes to a base document and can be thought of as an extension to that base). The easiest way to define a type 68 user record is to establish your own x9 rules specification, with your type 68 user record, as an override to one of the x9 specifications that are distributed with X9Assist.

This overall process consists of the following tasks:

- 1) Use an editor to create your x9 rules document.
- 2) Use the Configuration Editor to create your own configuration that uses your x9 rules document.
- 3) Test your configuration.
- 4) As desired, use Make to create your type 68 user record.

The following x9 rules document can be used as an example of an x9 override document that includes a type 68 user record override.

```
<?xml version="1.0" encoding="UTF-8"?>
<x9rules>
  <x9Controls>
    <x9Specification>UCD</x9Specification>
    <characterSet>EbcDic</characterSet>
    <maximumFileSize>2048</maximumFileSize>
    <fieldZeroPresence>required</fieldZeroPresence>
    <fieldZeroFormat>bigEndian</fieldZeroFormat>
    <multipleLogicalFilesAllowed>false</multipleLogicalFilesAllowed>
    <iclCollectionTypes>=00|01|02|12</iclCollectionTypes>
    <iclRecordTypeIndicators>=E|I|F</iclRecordTypeIndicators>
    <iclrCollectionTypes>=03|04|05|06</iclrCollectionTypes>
    <iclrRecordTypeIndicators>=E|I|F</iclrRecordTypeIndicators>
  </x9Controls>
  <basis>
    <base>x9rules_x9.100-187_UCD.xml</base>
  </basis>
  <overrides>
    <x9record>
      <type>68</type>
      <format>100</format>
      <name>User Record</name>
      <length>f65</length>
      <location>=a25g</location>
    </x9record>
  </overrides>
</x9rules>
```



```
<field> <item>x68.01-p001-l002-mandatory-notModifiable</item>
  <edit>n</edit>
  <name>Record Type</name> </field>
<field> <item>x68.02-p003-l001-mandatory-modifiable</item>
  <edit>an</edit>
  <values>=5</values>
  <name>Owner Identifier Indicator</name> </field>
<field> <item>x68.03-p004-l009-mandatory-modifiable</item>
  <edit>ans</edit>
  <values>=ITEM</values>
  <name>Owner Identifier</name> </field>
<field> <item>x68.04-p013-l020-conditional-modifiable</item>
  <edit>ans</edit>
  <values>=USER RECORD</values>
  <name>Owner Identifier Modifier</name> </field>
<field> <item>x68.05-p033-l003-mandatory-modifiable</item>
  <edit>an</edit>
  <values>=100</values>
  <name>User Record Format Type</name> </field>
<field> <item>x68.06-p036-l003-mandatory-modifiable</item>
  <edit>n</edit>
  <values>=001</values>
  <name>Format Type Version Level</name> </field>
<field> <item>x68.07-p039-l007-mandatory-modifiable</item>
  <edit>n</edit>
  <values>=20</values>
  <name>Length of User Data</name> </field>
<field> <item>x68.08-p046-l019-mandatory-modifiable</item>
  <edit>an</edit>
  <name>Card Number</name> </field>
<field> <item>x68.09-p065-l001-mandatory-modifiable</item>
  <edit>n</edit>
  <values>=0|1|2|3|4|5|6</values>
  <name>Authorization Code</name> </field>
</x9record>
</overrides>
</x9rules>
```

Using Excel Formulas

Assuming your use case file is maintained using Excel, you can enhance your process by using formulas to provide further validation of your user record fields. You can use formulas to validate your fields, highlight errors, or assign values based on other fields that are present.

Make Support for the Type 68 Record

Make can be used to easily create type 68 user records, which otherwise can become a complex process.

Each Make reformatter can be associated with a single type 68 user record. This association is established using the “type 68 user record” tab within the Make user interface. Once you establish the link to your type 68 user record, you can save your reformatter. The link is saved allowing your reformatter to reference your user record in all future X9Assist sessions.

Similar to other fields supported within the Make process, you can point Make to a specific column within your use case file for each field within your type 68 user record.

- The type 68 data can be provided for either debits and/or credits.
- Each type 68 data field can be supplied from a particular column within your use case file, or can be assigned a constant value.
- A type 68 record will be created whenever the type 68 data field is not spaces.

XML Support

X9Assist supports type 68 user data in either traditional form (which consists of a series of user defined fields) or as an embedded XML document (where the user data is created as an XML document consisting of the user defined fields). Both of these formats are common and utilized within the industry.

XML support is activated at the type 68 record type level using the “xmlTags” parameter, which provides basic information needed to create the user XML document. The format of the “xmlTags” parameter is as follows:

```
<xmlTags>DocumentName|FieldCountTag|FieldGroupTag|FieldTag|NameTag|ValueTag</xmlTags>
```

Each type 68 user field must then have its corresponding “xmlid” , which is the tag that will be used to create the individual XML field.

The following is an example of a type 68 user record defined as an embedded XML document.

```
<?xml version="1.0" encoding="UTF-8"?>
<x9rules>
  <x9Controls>
    <x9Specification>UCD</x9Specification>
    <characterSet>EbcDic</characterSet>
```

```

<maximumFileSize>2048</maximumFileSize>
<fieldZeroPresence>required</fieldZeroPresence>
<fieldZeroFormat>bigEndian</fieldZeroFormat>
<multipleLogicalFilesAllowed>false</multipleLogicalFilesAllowed>
<iclCollectionTypes>=00|01|02|12</iclCollectionTypes>
<iclRecordTypeIndicators>=E|I|F</iclRecordTypeIndicators>
<iclrCollectionTypes>=03|04|05|06</iclrCollectionTypes>
<iclrRecordTypeIndicators>=E|I|F</iclrRecordTypeIndicators>
</x9Controls>
<basis>
  <base>x9rules_x9.100-187_UCD.xml</base>
</basis>
<overrides>
  <x9record>
    <type>68</type>
    <format>100</format>
    <name>User Record</name>
    <length>v82</length>
    <xmlTags>MyXmlRecord|Count|Field|name|value</xmlTags>
    <location>=a25g</location>
    <field> <item>x68.01-p001-l002-mandatory-notModifiable</item>
      <edit>n</edit>
      <name>Record Type</name> </field>
    <field> <item>x68.02-p003-l001-mandatory-modifiable</item>
      <edit>an</edit>
      <values>=5</values>
      <name>Owner Identifier Indicator</name> </field>
    <field> <item>x68.03-p004-l009-mandatory-modifiable</item>
      <edit>ans</edit>
      <values>=ITEM</values>
      <name>Owner Identifier</name> </field>
    <field> <item>x68.04-p013-l020-conditional-modifiable</item>
      <edit>ans</edit>
      <values>=USER RECORD</values>
      <name>Owner Identifier Modifier</name> </field>
    <field> <item>x68.05-p033-l003-mandatory-modifiable</item>
      <edit>an</edit>
      <values>=100</values>
      <name>User Record Format Type</name> </field>
    <field> <item>x68.06-p036-l003-mandatory-modifiable</item>
      <edit>n</edit>
      <values>=001</values>
      <name>Format Type Version Level</name> </field>
    <field> <item>x68.07-p039-l007-mandatory-modifiable</item>

```

```
<edit>n</edit>
<values>=20</values>
<name>Length of User Data</name> </field>
<field> <item>x68.08-p046-l019-mandatory-modifiable</item>
<edit>an</edit>
    <xmlid>CardNumber</xmlid>
    <name>Card Number</name> </field>
<field> <item>x68.09-p065-l001-mandatory-modifiable</item>
<edit>n</edit>
<values>=0|1|2|3|4|5|6</values>
    <xmlid>AuthCode</xmlid>
    <name>Authorization Code</name> </field>
</x9record>
</overrides>
</x9rules>
```

X9.37 Extended File Editing

X9Assist has extensive capabilities to create and modify x9 files. However, there are times when you want to edit an x9 file at the individual record level for specific conditions. Examples are:

1. Create duplicated items for specific use cases.
2. Create various errant file conditions (duplicated headers, duplicated trailers, etc).
3. Rearrange the sequence of records within a bundle or cash letter.
4. Cut and paste items from one x9 file to another.
5. Move a type 61 credit record from one position to another within a file.
6. Insert a type 61 credit into an x9 file when missing.

X9Ware LLC has collaborated with SPFLite on the ability to use their editor against x9 files. SPFLite is a shareware tool with a line editor that mimics the IBM mainframe SPF editor. This x9 editor functionality represents an important tool for your x9 toolbox and opens new possibilities that are otherwise not possible. There are numerous file editors that are available within today's marketplace. We reference SPFLite because it is a proven tool, implements a common user interface, has extensive functionality, includes profiles which support the unique requirements to parse x9 files at the record level, and is immediately available for your download and usage.

We provide SPFLite as an example. You are encouraged to review available tools and select one that meets your specific needs. Please be aware that the ability to edit x9 files require SPFLite Build 7.1.4050 or higher. This build level is needed to support the x9 field zero (length) separators in big endian format. There are many earlier versions of SPFLite on various shareware download sites, so you need to double check that you have a build level that contains the needed capabilities.

SPFLite

If by chance you are familiar with the SPF editor under IBM TSO, you will welcome the user interface since it dramatically mimics that user interface. You will be up and running immediately.

SPFLite has an extensive help facility which can be invoked either by the Help command or by hitting the F1 key. It is recommended that you spend some time reading through that material, which will allow you to fully understand that editor functions.

SPFLite is a full screen editor that works especially well with block commands. For x9 files, these are very powerful functions since they allow you to very easily move, copy, replicate, and delete entire blocks of records using one or two commands.

For more information, please reference our white paper which is available on our website (www.x9ware.com), the SPFLite website (www.spflite.com), or contact us for more information.

ACH Configurations

The X9Assist validation engine is based on

ACH Configuration Name	XML file name	Description
nacha-2013	achrules_nacha_2013.xml	Implements our understanding of the NACHA 94 byte ACH standard.
nacha-core-validations	achrules_core_validations.xml	An extension of nacha-2013 that is a bit more lenient than the formal standard, and is designed to represent the ACH standard as used by many processors and financial institutions. We welcome your comments and feedback as to how we can make this better. This extension will not be perfect for everyone, but our goal is to maintain the spirit of the NACHA specification while reducing meaningless errors.
nacha-site-validations	achrules_site_validations.xml	An extension of nacha-2013 that is designed to be customized for any given site. We provide a default that you can then tailor to your specific needs. You can use this configuration as the basis and framework to implement a rule set for your environment.
nacha-no-validations	achrules_no_validations.xml	An extension of nacha-2013 that can be used to validate the integrity of an ACH file without application validation rules at the field level.

Customization of Nacha-Site-Validations

It may be desirable to implement a customized set of NACHA rules to validate files per your specific requirements. Please be advised that this is an advanced topic, and that X9Ware offers consulting services to help when needed. Generalized steps are as follows:

- 1) Review the “X9 Rules” section in the X9Ware SDK User Guide, which describes how our rules are implemented and the standard system edits that are available.
- 2) Review the XML definition of our implementation of the standard NACHA rules, on a record and field level basis. This XML file is available as file “achRules_nacha_2013.xml” in the program installation folder, which for example might be / Program Files / X9Ware LLC / X9Assist Rxx.xx / rules / x9rules / .

- 3) Run various file validations for your user submitted files and review errors that are currently being thrown and build a list of those fields and errors that are targeted for change. This step is crucial, since it forms the basis for all subsequent work.
- 4) Review the list of the field and validation changes that you have identified as desirable. Research each of the field level errors to fully understand what they mean and why they are being thrown. Ensure that each error you want to remove or relax is appropriate.
- 5) Finalize your list of field level rule changes. A given change might reduce the severity of the existing error that is being thrown (eg, from error to warning, or error to informational) or it might eliminate the error completely.
- 6) Review our sample “achrules_site_validations.xml” file, which will serve as a basis for your development work. You will find this file in your X9Assist installation folder (for example, in Program File) in folder / rules / x9rules / .
- 7) Copy the “achrules_site_validations.xml” file into your folder / documents / x9_assist / rules / x9rules / . You will be making your changes to this copy and NOT to the original file that is located in the program installation folder.
- 8) The next step is to use an XML editor editor (such as NotePad++) to make your changes to the XML definition. If you are using NotePad++, we also recommend that you use their XML Plugin, since it will validate your XML tags to ensure they are matching pairs. An alternative is to use the X9Assist Xml Editor. X9Ware has written this tool as our future direction to make this process as easy as possible. However, our editor is experimental, so be advised of that current status. We would appreciate your feedback regarding the use of our editor.
- 9) Once your XML rules have been constructed and saved to / documents / x9_assist / rules / x9rules / , they are immediately available to X9Assist. You must first load an ACH file, and then test your validation rules by using “Revalidate Using” on the tool bar. Within this drop-down combo box, select entry “nacha-site-validations” to launch your customized validation rules.
- 10) This will become a repetitive process, as you make changes and then retest.
- 11) You can contact us for possible consulting assistance if you have specific validations that you have been unable to accomplish, or run into barriers that you cannot work through.
- 12) Once you complete your changes, you can share your customized XML file to other users within your environment, allowing them to take advantage of your work.

Default Nacha-Site-Validations XML File

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<x9rules>
  <copyright>X9Ware LLC 2012-2018</copyright>
  <company>X9Ware LLC</company>
  <release>R5.04</release>
  <buildDate>2022.07.04</buildDate>
  <timestamp>20220706_074541_761</timestamp>
  <basis>
    <base>achrules_nacha_2013.xml</base>
  </basis>
</x9rules>
```

```

    <gender>ACH</gender>
    <dialect>ACH</dialect>
    <x9Specification>nacha site validations</x9Specification>
    <characterSet>either</characterSet>
    <lineBreaks>optional</lineBreaks>
    <fieldZeroPresence>prohibited</fieldZeroPresence>
    <maximumFileSize>0</maximumFileSize>
    <dateMinimumYear>1993</dateMinimumYear>
    <dateMaximumYear>2099</dateMaximumYear>
    <dateWindowMinusDays>1095</dateWindowMinusDays>
    <dateWindowPlusDays>95</dateWindowPlusDays>
    <userFieldsValidated>true</userFieldsValidated>
    <reservedFieldsValidated>true</reservedFieldsValidated>
    <requiredFieldsAreMandatory>false</requiredFieldsAreMandatory>
    <multipleLogicalFilesAllowed>false</multipleLogicalFilesAllowed>
    <creditBalancing>single</creditBalancing>
    <creditOrientation>any</creditOrientation>
  </x9Controls>
  <overrides>
    <x9record>
      <type>1</type>
      <format>000</format>
      <name>File Header Record</name>
      <length>f94</length>
      <field>
        <item>x01.04-p014-l010-mandatory-modifiable</item>
        <name>Immediate Origin</name>
        <justify>r</justify>
        <edit>achImmediateOriginNoMod10</edit>
      </field>
    </x9record>
    <x9record>
      <type>6</type>
      <format>CTX</format>
      <name>CTX Entry Detail Record</name>
      <length>f94</length>
      <field>
        <item>x06.07-p040-l015-optional-modifiable</item>
        <name>Identification Number</name>
        <edit>ans1b</edit>
      </field>
    </x9record>
  </overrides>
</x9rules>

```


Branding

The Enterprise licensing option allows a branded version of our desktop products to be provided to your client base. This branded version is a white label copy of our tools that could be packaged and provided to your customer base.

This option provides a real benefit during the on-boarding of your new clients, since they will be able to run their own initial file validations prior to running their first integrated test into your image exchange application. This functionality can then be used by your clients on an ongoing basis for testing and research.

In this mode, our X9Ware desktop product is branded with:

- Corporate name
- Corporate logo (two can be provided which will be displayed in the thumbnails).
- Descriptions of your products and services on the About panel
- References and URL linkage to your website

X9Ware will be glad to work with you on the requirements, design, and implementation of this solution. You will find pricing that is extremely beneficial to you, making this a very low cost solution. X9Ware has taken that approach since we want to build relationships, and we see this as an excellent way to work directly with you on such a broad based solution.

FAQ

Number	Question	Answer
1)	Why would we select the X9Ware desktop tools over similar products?	Compared to the competition, our desktop tools offer the highest levels of capability, flexibility, and performance at the lowest cost. The user interface is designed to be intuitive and easy to use. X9Assist has a large number of testing and data generation features. Our desktop products continue to be enhanced, and we would like to hear your suggestions as to how we can make them better.
2)	How can X9Ware support both the x9 and ach file formats?	Our initial x9 implementations were based on a record and field design was very generic in nature and easily extended from the x9 data format to the ach data format. It was a logical extension and fit nicely within all of our tools, and prepares us to support additional file types in the future. This core design allowed our viewers and validation tools to be easily enhanced to support the ach standards.
3)	How does the licensing of dual support of x9 and ach work?	All of our desktop tools (from our freeware viewer and up) all support basic browse functions of x9 and ach files. This is “built in” as core capabilities. Users then decide if they want to license more enhanced functionality for x9, for ach, or for both.
4)	Are there separate desktop programs for x9 versus ach?	No, they are built into the same executable. When you launch a file, there is an automatic inspection performed which determines if the file is x9 or ach.
5)	Are there separate desktop programs for lite versus validator, assist, and so forth?	No, they are build into the same executable. The functionality that is extended to the user is based on the license type. Not only does not simplify things on our end, but it also makes it very easy for users to change to different license types.
6)	What monitor sizes are supported by the X9Ware desktop tools?	The absolute minimum monitor size is 1024 x 768 with built-in functionality to take full advantage of increasing large monitor sizes. An internal preferences file is updated based on your window and frame sizes, your split panel resizes, and your usage patterns. You can always revert to standard panel sizes by deleting the GUI preferences file that is created in folder / documents / x9_assist / xml /.
7)	What file name extensions are used to identify files within my input folders?	File extensions can be separately defined for x9 and ach via program options, where you can explicitly define the extensions that are assigned to each file type (x9 versus ach). Out of the box, these are defined as x9, x937, icl, and dat as x9 file extensions, and then ach as the nacha/ach file

Number	Question	Answer
		extension.
8)	Can I set the default folder to be used when I open a new file?	Yes, a variety of default folders can be set via program options.
9)	Is there an easy way to resize frames and panels?	Yes, if you look in the lower right hand corner of most frames, you will see a “handle” that can be used to adjust the frame size using your mouse.
10)	Not that we expect problems, but how do I report problems and how fast will they be resolved?	We apologize in advance for any problems that you encounter. We have tested extensively but also realize that there is a lot of complexity within the x9 and ach environments. We are very confident that we can resolve virtually any problem within 5 days. If you encounter a problem, send us a short description of what you attempted to do, what went wrong, and if appropriate attach your system log for this session. The system log is very helpful since it provides trace information on the classes and methods where the error occurred. Please send your communications to x9assist@x9ware.com .
11)	When I launch a file, how can I tell when all processing and validation has been completed?	You can do this in several ways. Watch the progress bar that is located in the lower right, where you will see progress as a percentage of the work being done. You can also watch the status bar which is located at the bottom of the main panel. In the lower left corner, you will see a “runner” icon while your file is being processed. Once validation is complete, the runner icon will change to a final icon which represents the severity of any errors that were found.
12)	Are there any keyboard “hot keys” that can be used when browsing the currently loaded file?	There are no globally defined keys that can be used for browsing, since keys are instead defined locally to individual panel. You can easily browse through batches and items using the arrow buttons that are provided on the tool bar. You should also visit the navigation instructions that are provided as a tab on the viewer tree.
13)	Can we customize maximum file size (in MB), maximum checks per file, maximum checks per cash letter, and maximum checks per bundle?	Yes, these can all be set via program options. Errors are automatically generated when those limited are exceeded.
14)	Can we customize minimum and maximum	Yes, these can all be set via the file validation rules.

Number	Question	Answer
	image size in bytes, minimum and maximum size in inches, and total image size (front + back)?	
15)	How can I easily review all of the errors in a given file?	<p>There are numerous ways to do this. A combination of these techniques will work the best for you, based on the situation at hand.</p> <p>Within the tabbed panel, the “Summary” tab provides a summary of all errors that were encountered. This is by field number, which allows you to focus directly on the errors that were identified.</p> <p>With the tabbed panel, the “Errors” tab will similarly show you each type of error that has been identified. Within this tab, you can expand and collapse the detail behind any given error. While reviewing this information, you can also launch directly to any given error within the currently loaded file, allowing you to see all of the fields in that data record and the overall context of the error.</p> <p>The “Error Viewer” tree on the left is a list of just those items that are marked in error. You can use this tree to walk through the errors one at a time, either using the arrows on the tool bar or their corresponding shortcut keys.</p>
16)	We are getting errors due to requiring blanks in user and reserved fields. Can these edits be disabled?	Yes, this can be done through program options on the “validation” tab.
17)	How can I export a list of all of the errors for a given file?	<p>There are several ways to do this.</p> <p>One easy way is to use the export errors function, which is invoked using export from the tool bar. First run validations against your file. Then select export from the tool bar and next select the “Export Errors” tab. You can then indicate which errors you want to export (all, data related, or image related). You can initiate using the export button, select your output file, and then generate a file that contains all errors. The output file will be in CSV format which can be pulled into Excel or compatible tools.</p> <p>There are several other ways to do this. Both print and Excel export will generate lists of all errors. These facilities have advantages because you can incorporate other</p>

Number	Question	Answer
		<p>information into those extracts, such as the file attributes, cash letters, bundle totals, and so forth.</p> <p>The Excel Exporter is especially helpful and the output from this function is in a format that is easily viewed and shared with others. This tool has the ability to export information at the overall run level (the entire file) or the current record group that is being displayed in the browser. You can launch the Excel Exporter from either the tool bar or the menu bar.</p>
18)	I have run a validation against a file and there are errors on a particular field that I need to share with others. What is the best way to show exactly what is wrong with the record and field in question?	The Excel Exporter is the best way to do this. Position in the viewer on the particular record that has the error and then run the Excel Export from the tool bar. First uncheck “all run level tables” and then check “all record level tables”. This will create an excel file that documents the exact error(s) that have been identified in this specific record group. This XLS file is located in the /documents/x9_assist/reports folder and can be shared with others to describe the identified errors.
19)	We would like to use the Excel Exporter interface, but not all of our workstations have MS Office installed. Are there options?	Yes, you can also Libre Office which is an open source tool.
20)	How can I get a list of all record types that are present in a given file?	As part of file validation, a summary of all of the record types is generated. This list is displayed in the record types tab and includes the number of errors associated with each record type.
21)	How can we tell what edits are being applied to each field?	From the DashBoard, look at the Field Viewer tab. If you slide this to the right, you will see columns that define each edit that is being invoked for each field. You will note that a single field can have multiple edits.
22)	Can the list of valid values for a given field be changed?	Yes, if there is a very specific reason to do this. Please let us know if our installed validation rules are incorrect and require modifications.
23)	We would like to print on legal size paper. Does print support that?	Not currently, but you can still do this by using Excel Exporter and then printing on legal from there. This provides the advantage that you can totally control your print format, highlighting, column headers, and so forth.
24)	Can I convert a file from ASCII to EBCDIC, or can	Yes, you can change these file attributes when you Save a new file.

Number	Question	Answer
	I create or remove field zero lengths?	
25)	Is Adobe Reader required for PDF viewing?	This was a requirement with earlier releases of our desktop products, but there is no longer any such base requirement. Instead, help files and reports are viewed using your installed internet browser. (they are HTML files).
26)	How can I force a file to be analyzed using x9.100-187 and not the x9.37 rules?	This can be done on a one time basis (for just the file currently being analyzed) on the toolbar. This setting is “sticky” and will be remembered from session to session. This , or this rules assignment can be made permanently.
27)	How can we validate origination and destination ABA in the file header, and not just accept anything populated in those fields?	This validation is optional. You can enable ABA validation for the file header using the ABA Editor, where you can define a list of ABAs and then indicate which are originators, which are destinations, and so forth. Out of the box, this validation is disabled through the use the generic entry (9999999999) which is a wild card that represents all financial institutions.
28)	We have different validation rules for various types of files such as In Clearings, POD, Remote Deposit, and Branch Capture. How do we get the correct validation to be run automatically?	As stated above, you can override the rules for a single run by setting your Configuration. More complex situations can be automatically controlled through use of the Binder. This requires that you create a logical configuration for each unique validation scenario. Each configuration will be assigned a name and will consist of rules, tiff rules, messages, and options. These configurations are defined and maintained using the Configuration Editor. You can then use the Mapping Editor to define rules that will select the appropriate configuration based on fields from the file header. This allows the binder to automatically select the correct configuration and rules to validate any given file.
29)	How can I export a single image from an x9 file, and get the exact TIFF image that is present in the x9 file? Can this be done?	Yes. Find and display the check in question. Then launch the Item Viewer (this is the magnifying glass on the tool bar). Within the Item Viewer, you can then export the image to an external file. Note that you can do this for either the front or back image. If you only need the image as an attachment, an easier way is to copy the image to the system clipboard, which is another function available in the Item Viewer. With the image in the clipboard, you can then paste the image in any application wherever needed.
30)	How can I tell what tiff tags are on a given image?	Use the browser to position on the item in question. Once you have located the item in the viewer, then launch the Item Viewer and select the “Show Tiff Tags” check box in the action panel at the bottom. The Item Viewer allows you

Number	Question	Answer
		to toggle between the front and back images for any given item, as well as through the gray scale images if they exist within your file. You will see the tiff tags in a table in the lower half of the display. If you still have questions about your image, then you can export the image (using the previously documented procedure) and then use another tool such as AsTiffTagViewer to look at the tiff tags.
31)	What is EOFB and can that validation be disabled?	EOFB is End of Facsimile Block and is part of the Group 4 Fax standard. EOFB appears at the end of each segment within the compressed tiff image. This can be enabled or disabled via the “validateEOFB” xml parameter.
32)	What is multi-strip and can that validation be disabled?	Multi-strip was designed for TIFF viewers to reduce memory requirements when attempting to display very large images. This is not an issue for the small black and white images that are used per Check21 standards. Financial institutions have defined exchange standards that require single strip images. Yes, multi-strip validation can be disabled per the XML tiff rules.
33)	I now know that a specific file contains errors and I need a list of all records with a certain error type so we can pursue the issue with the originator. How can I get such a list?	Create a filter and indicate you want errors only, and then select the error type that you are interested in. You can then get the list using either Excel Exporter or Print.
34)	After I have made a lot of changes to a file, there are several changes that I need to reverse but I do not recall the original value. What is the easiest way to undo selective changes that I have made?	Modify maintains a log of each change that you have made to the current file. That log includes a “revert” button that you can use to remove your field modification by reverting to the original field value.
35)	Can Modify be used to remove records from a file?	Yes, the delete function has a variety of options. You can remove single records, ranges of records, individual items, bundles, and so forth. Delete is a function that runs with Modify enabled.
36)	How can I get a list of all items in a cash letter?	Within the tabbed panel, the “Items937” tab contains a list of all bundles within the file, where each bundle can be expanded. In the column headers, there is also an “Expand”

Number	Question	Answer
		checkbox which will expand all bundles. This list of items can be then be processed by Print or the Excel Exporter. This approach allows you to selectively get a list of only those bundles that interest you, or if you desire, the entire file.
37)	I am trying to find and filter on a specific item but cannot find it. What can I do to improve my search?	<p>Make sure you are taking advantage of all of the information that you have. For example, if you provide the check serial number, Find will look search for that in both the Process Control and Aux Onus fields.</p> <p>If you are still unable to find the item, one method to allow you to drill down into the data is to create a filter on all items using “select all”. Once you create that filter, you can then click on the column headings and sort on any of the provided columns. For example, you can sort on amount, sequence number, ABA, MICR OnUs and Aux OnUs. Once you have sorted on any given column, you can then forward through that list looking for your item. This approach sometimes helps to find an item which otherwise is difficult to locate.</p>
38)	How can we remove a small number of images from an x9 file that has identified issues?	<p>Modify has the ability to replace individual images with a “missing” image document.</p> <p>The Modify function is enabled and disabled from the toolbar or the menu bar. Once modify is enabled, you can then modify the image field (52.19) in the type 52 record and indicate if you want to replace either one image or both images associated with the check with your missing image document. After all modifications have been completed, you should then disable modify, run a validation from the tool bar, and then write a new output file.</p>
39)	Can we customize the missing image document?	<p>Yes, this document is stored in the program files images folder. The document can be modified using a typical paint program (GIMP is open source and is excellent). Remember that the missing image document must remain as black/white.</p>
40)	If I am using Modify to make changes to a file, how can I get a list of all of the fields that have been changed?	<p>We realize the importance of this documentation. Modify maintains a log of each change that is made to the currently loaded file. This includes the original value that was in the file and your modified (new) value. You can see this list on the Modify panel, save it to an external file during the save process, and you can get a hard copy through either Excel Export or Print.</p>
41)	We want the Use Case	Yes, you can create your own ABA lists and then utilize in

Number	Question	Answer
	Editor to create lists that can actual bank ABA numbers and not the generic lists that are manufactured based on abaList.csv. Is there an easy way to accomplish that?	our generation functions. We also have an aba list that represents all US banks and thrifts and will provide that list to financial institutions and other appropriate users on their request.
42)	We want to use Generate to create test files using our account number ranges, but we do not have use cases from a test system. How do we work around this issue?	<p>There are several ways. Generate will randomly select account numbers from a use case file that you can easily create using the Use Case Creator. You can utilize a generic (pre-generated) list, or you can create new use case files using your account number ranges and “Modcheck” specifications.</p> <p>Another even easier way to do this is from an existing file. Granted that this uses production information (if you are using a production file), but it will be limited to production account numbers (with their associated ABA and check numbers). The remainder of the file that is generated will be randomized non-production data, including the images. The advantage is that this approach is very little work and will generate items with MICR lines that match your environment. Export also has the ability to merge into an existing account number list, so you can extract from multiple files and build an accumulated list.</p> <p>A similar way to accomplish this is to use an existing OnUs file to export the type 25 records to a CSV, which can then be used as input to Generate.</p>
43)	We like the account number list that is used by Generate and Scrub, but we want to use the actual use cases from our test system. Can we incorporate our list into the process?	Yes. Take a look at the simple Excel format that is used for account number lists. There are four fields (account, ABA, process control, and auxiliary OnUs). You can extract this information from your test bed, or you can utilize Excel (or some other tool) to create and maintain the account number list. Once you have prepared a list, you can then store it in the “lists” folder where you it can be utilized by Generate and Scrub. Note that these functions will randomly select from this list. You need to make sure that the list is substantially larger than the number of items you will want to generate in a single run. Refer to our documentation on the Use Case Creator which includes more detailed directions on how to create your own account number list.
44)	We are currently testing	Yes, there are several ways to do this. One way is to use

Number	Question	Answer
	with production data and our Auditors have taken exception to that process. We do not want to go as far as using Generate to create random files. Is there another approach that will still protect customer data but stay close to our production file contents?	Scrub against one of your existing files. This would allow you to retain all existing data and only replace the images with dynamically drawn data. This removes a lot of potential customer data that is present on the check images. Generate will dynamically draw checks and will retain the amounts and the MICR lines. This is an important step towards meeting your Audit requirements.
45)	How can I reset to default program options?	There are several ways to do this. First and easiest is to use the reset function that is available in program options. You must reset and then save. Another way is to just delete the programOptions.xml file.
46)	We have made simple value changes to xml and tiff rules. How do we reconcile those when we install a new release? And are there other issues that we need to be concerned about when we install a new release?	<p>Our installation process will remove various components of the old release before it installs the new one. However, rules are identified as system versus user, with the intent that only the system rules are automatically replaced. Our intent with this design is to ensure that user created rules are not replaced when a new release is installed.</p> <p>It is our recommendation that you do not actually update the data/tiff rules as provided by X9Ware, but that you instead apply your changes to copied files within the configuration definitions.</p> <p>We recommend that you install and use a tool such as WinMerge, which will compare files and highlight the differences between two versions of the same file. This is an example of many such tools that are available.</p>
47)	We have a license that supports multiple users. Can we install such that certain functions (such as modify) are disabled for our general user base?	Yes. A profile facility which defines which functions are allowed within a given environment. The profile is stored in the program launch folder, where a typical user will not have the authority to make modifications. You can internally package and distribute these profiles to limit user functions as needed.
48)	We want to use Modify, but we want to restrict which fields can be modified. Can that be customized?	Yes, this can be done through the configuration rules.
49)	We have a license that	Yes. You can have your administrator enter your license key

Number	Question	Answer
	supports multiple users. Can we install such that a single user can be responsible for the license key?	which will create a license.txt file in folder /documents/x9_assist/license. You can then distribute this license file to required users. If you are running X9Assist from a network drive, then you can place this license file in your /Program Files/X9Ware LLC/X9Assist/license folder, and it will be automatically distributed to users whenever updated based on file modification date.
50)	We have a license that supports multiple users. Can we install such that our data and tiff rules are maintained centrally and automatically distributed to users?	No. You would either have to do your own distributions of the data/tiff rules, or you would have to run X9Assist from a network drive which would allow the rules to be administered centrally.
51)	How do I add new fields or entire records to the rules definitions?	<p>Please feel free to reach out to us to work with you on this, where this support is provided as part of our Extended Support agreement. One of the advantages of our technical design is that new fields and even new record types can be added; in most situations these are XML updates only. However, there are times that new edits must be internally added to our product to support your requirements, and there are always a lot of questions during this level of customization. Added new record types is more complex depending on exactly how they are to be handled. Hence we require an Extended Support agreement to provide the funding that will be needed.</p> <p>Similarly, our data/tiff rules are fully documented and that information is provided as part of our Extended Support agreement.</p> <p>Your updated rules should be updated in the installation folder (eg, Program Files). X9Assist will automatically copy these definitions to your user runtime folder the next time that X9Assist is run.</p>
52)	Can we change the handwriting font that is used to draw checks?	Yes, there are several alternatives provided, and many fonts are available at either no cost or very lost cost. The handwriting font used in the various check formats is specified via documents that are stored in folder /documents / x9_assist /xml / checkFormats / .
53)	Can we change the MICR font that is used to draw checks?	Yes. Please contact us if for more information on how to use your own MICR font with X9Assist.

Number	Question	Answer
54)	Can the basic customer and bank information that is used to draw checks be changed?	Yes, these fields can be customized via program options on the “draft check” tab.
55)	Can we create new check formats with our own images?	Yes, you must create your image template and place it in the /Program Files/X9Ware LLC/X9Assist/images folder, and then update /documents/x9_assist/xml/checkFormats.xml with your format definition. You can review one of the existing check formats and follow that as a guideline. Note that the x/y coordinates of each check field must be specified in inches. Those coordinates can be obtained using a paint program such as GIMP, which will provide the coordinates by pointing at each field with your mouse. X9Assist requires that the image be stored in the /Program Files/X9Ware LLC/X9Assist/images folder at a 240 DPI resolution.
56)	Can we get a list of the actual data records exactly as they appear in the input file?	This question obviously applies to x9 only. Yes, you can use Export to get a list of the raw data records which you can analyze or route to other applications. One use of this file is to bring it into a utility that supports REGEX which would allow complex searches.
57)	Can we get a list of all possible error messages?	Yes, these are defined in the program files rules / messages folder.
58)	Can the severity of individual error messages be changed? Can an error be reduced to warning or informational?	Yes, the error severity for each message is sent in the program files rules / messages folder.
59)	Can we get offline access to the help documents?	Yes, these are combined into an X9Assist User Guide which can be downloaded from our web site.
60)	Can we author our own reference documents to further enhance the help topics?	Yes, you can add your own help files within the program files help folder. We write our help topics using Libre Office and then export content to html for viewing.
61)	Is this source code escrowed by X9Ware?	Software Escrow is available from Iron Mountain and we encourage all customers who have this requirement to contact us to establish this third party agreement.
62)	How do we suggest editing and usability improvements?	Please send them to us at x9assist@x9ware.com . We are very open to these, since we want to make this product the very best it can be. If you have requirements, then we suspect that others will have something similar. Our approach will be to generalize the requirement so that it is

Number	Question	Answer
		usable by as many clients as possible.
63)	How do I determine when my license expires, and will I get any warning?	You can launch the Registration Editor from the menu bar and it will show licensing information including the expiration date. X9Assist will issue popup warning messages in advance of that date. You can renew you license early and at any time, and still get the full future dated expiration date.
64)	Can I upgrade my license, for example, from Solution Pack to Unlimited?	Absolutely. We will issue a new license and give you pro-rated credit towards your expanded license.
65)	We are interested in the SDK. Is there a run time license key?	Good question and the answer is no. Our SDK has significant advantages over our competition since they have license keys in their SDK that will expire. We see that as a time bomb that is lying in wait. Our approach is to instead provide a code word that activates the SDK, without any time out. We see this as our honor system and know that we will both appreciate this approach, which eliminates the potential for things to arbitrarily stop working at Midnight on some day in the future.

Your suggestions for improvements to this FAQ are always welcome !