X9Ware LLC

White Paper
On
X9 File Testing

Revision Date: 08/25/2022
Release R4.10

X9Ware LLC
10753 Indian Head Industrial Blvd
St Louis, Missouri 63132-1101
(844) 937-1850

Email: sales@x9ware.com

Problem Statement

Building quality systems has always required robust capabilities to quickly and easily manufacture test data for the specific conditions that must be tested. Test data must be targeted at the specific conditions that are represented by individual test cases.

X9 (image exchange) application testing has been complicated by the fact that test files must have their "data" and "image" components created in a fully correlated fashion. However, it is difficult to generate such files, since the creation of the check images is a complex problem. As a result, many organizations have reverted to creating paper tickets (using encoding machines) that are then captured and used to prepare the required test X9 files. This is both a complex and labor intensive operation. Other organizations have indicated that they cannot directly solve this problem and instead utilize production files for their testing, to the dismay of the management and audit teams.

A very effective way to test any application is to build a repository of test conditions which have an expected result. Although this is also complex, the repository becomes a very effective tool for regression testing that can be applied as a standard measure when application changes are being made. In these situations, the repository is first run against a production version of the application and used to create a baseline. The repository can then be used in future tests and compared to the baseline. The repository should be viewed as a "living" set of test cases that would continue to be expanded and improved as part of application maturity.

Performance testing requires a large amount of appropriate data to push the application to a desired stress level. The test data must be relevant and must high volume. In accordance with management and audit requirements, this should not be "production" data but should instead by sanitized to ensure that customer data is not compromised as a part of this testing process.

In summary, a fully functional X9 data generation tool must support the following types of functions:

- Creation of x9 files (data and images) from use case definitions
- Ability to build a repository of test cases which can be used for regression testing
- Ability to sanitize production files to easily create representative test files
- Ability to create high volume stress files from sanitized production data

Testing Phases

Let's first define a list of phases that are traditionally utilized within most testing frameworks. The overall phases required by successful x9 file testing is similar to any other business application. These testing steps will vary from organization to organization, but the overall intent is as follows:

Testing Phase	Description
1) Unit Testing	Initial tests of application components in a lower level environment to ensure their correct behavior prior to the initiation of higher levels of testing.
2) Functional testing	Functional testing will validate that an application conforms to its specifications and correctly performs all its required business and technical functions. This testing will apply a series of tests (test cases) which perform a feature by feature validation of behavior, using a wide range of normal and erroneous input data. This can involve testing of the product's user interface, APIs, database management, security, installation, networking, and external application interfaces. Functional testing is often performed on a cycle by cycle basis, where failures within each cycle are tracked and are then ensured to be resolved in subsequent cycles.
3) Regression testing	Regression testing is the process of running and validating all repository based test cases as a part of the testing process for each new application release. Regression testing provides a consistent and repeatable validation of every new application release. The purpose is to ensure that the implementation of new enhancements and the resolution of identified problems has not resulted in the introduction of new issues. Athough regression testing can be performed manually, it typically requires the creation of approved test cases for the application that are then tracked by a problem management system. The test case repository for the application can be grown over time as part of the application maturity process.
4) Integration testing	Integration testing will typically look at an overall suite of applications that are combined and tested on an end-to-end basis as a logical group. The goal of integration testing is to ensure that application interfaces and handoffs are operating per all defined specifications. These applications may be either internally or externally developed and supported. This testing is critical in complex environments where independent applications have been integrated to meet business requirements.
5) Performance testing	Performance testing is used to measure application scalability against defined non-functional requirements. The goal is to identify bottlenecks and limiting factors within the application as well as in other areas such as database servers, the network, middleware, third

	party products, and related downstream applications. Performance testing is required by business critical applications that demand high availability and reliability. Performance testing generally involves an automated test suite that can be used to simulate a variety of normal, peak, and high load conditions.
6) Acceptance testing	Acceptance testing is performed in the final stages of an overall testing process to verity that an application meets all business and technical requirements. Acceptance testing is focused on all aspects of the application and provides the final approval to proceed with the installation.

FFIEC Testing Standards

The FFIEC (supported by the FRB) clearly indicates that use of production data in test systems is either prohibited or requires appropriate protection methodologies to ensure data confidentiality is maintained. However, many organizations continue to test their x9 applications using production x9 data, and many times as the fundamental basis of their testing plan.

The FFIEC standards can be viewed here:

http://ithandbook.ffiec.gov/it-booklets/development-and-acquisition/project-management/project-management-standards

FFIEC standards are clear that use of production data is the least desirable approach to testing. Using manufactured data is preferred since it minimizes the potential escape of confidential from the testing process. Use of production data requires adequate controls to ensure customer data confidentiality is maintained at all times.

Quoted excerpts from the FFIEC stated policies are as follows:

What is the FFIEC InfoBase?

"The Federal Financial Institution Examination Council (FFIEC) is a formal interagency body empowered to prescribe uniform principles, standards, and report forms for the federal examination of financial institutions by the Board of Governors of the Federal Reserve System (FRB), the Federal Deposit Insurance Corporation (FDIC), the National Credit Union Administration (NCUA), the Office of the Comptroller of the Currency (OCC), and the Consumer Financial Protection Bureau (CFPB), and to make recommendations to promote uniformity in the supervision of financial institutions. In 2006, the State Liaison Committee (SLC) was added to the Council as a voting member. The SLC includes representatives from the Conference of State Bank Supervisors (CSBS), the American Council of State Savings Supervisors (ACSSS), and the National Association of State Credit Union Supervisors (NASCUS). Visit the Council's website for press releases and information on the mission and work of the Council athttp://www.ffiec.gov/.

The FFIEC Examiner Education Office created the FFIEC InfoBase, which is a vehicle that enables prompt delivery of introductory, reference, and educational training material on specific topics of interest to field examiners from the FFIEC member agencies. The IT Handbooks are updated and maintained electronically using the InfoBase vehicle."

FFIEC Testing Standards

"Management should establish testing standards that require the use of predefined, comprehensive test plans, end-user involvement, and documented test results. ***Additionally, testing standards should prohibit testing in production environments or with live data. If copies of live (customer) data are used during tests, management should ensure that appropriate standards exist to protect the confidentiality of that data. Management can use test data generators, which are software applications that generate representative***

testing data based upon predefined parameters, to develop appropriate testing data. *Numerous automated applications are also available that test program logic, functional operability, and network interoperability."*

X9 Testing Using Production Data?

Various arguments are often made as to why production data must be utilized for x9 testing. These arguments exist and persist in a large number of x9 testing environments. What are the realities associated with those arguments?

We test with x9 production data because …..	Situation
1) Captured account information must match the "systems of record" that are downstream from the x9 capture environment	This is a weak argument which goes against most testing methodologies, and is difficult to defend to either internal or external auditors. Standard testing methodologies require that test environments have a defined test bed which includes the account numbers and attributes that are needed to support the test cases. Testing is not a blind process where you simply run "everything". The use of production data may be an arguable position as participating data (one of several sources) for a particular phase of testing (eg, the Acceptance Testing phase) but it should not be defined as the sole basis for the entire testing environment.
2) We do not have any source for x9 cash letter files other than production itself; our only alternative would be to encode and capture paper transactions	This position does not take into consideration that tools do exist to support the creation of x9 test files. The FFIEC has recommended that organizations investigate and use test data generators to assist in the creation of test files. There are tools that will create test transactions, and there are additional tools that will then build x9 cash letters from the generated transactions.
3) It would be an overwhelming task to define and build a large number of test cases to represent our x9 check transactions	X9 testing tools can be used to extract test cases from either your test system or from your x9 production files.
4) It is a difficult technical task to create x9 transactions from individual test case definitions.	X9 Testing tools can be used to either create check detail (type 25) records or return detail (type 31) records from your test case repository.
5) Our application testing process requires that the x9 check transaction matches the associated image; even if we generate test transactions, there is no way to then generate the required matching images	X9 testing tools can be used to create the required front and back tiff images from the check transactions that are generated from your test cases.

We test with x9 production data because …..	Situation
6) X9 files have complex header and trailer records that are not easily created	X9 testing tools can be used to wrap your check transactions with all of the needed header records, trailer records, and control totals that are required to create a valid x9 cash letter file.
7) We want to test specific invalid x9 data conditions that have caused production problems in the past, so we have saved those production files and we always include them in testing	X9 testing tools can be used to modify fields on test transactions to generate any test condition that you require. X9 testing tools also exist that will allow you to export error transactions and then merge them into a consolidated cash letter which can then be utilized for exception testing.
8) We need to include code line data mismatch items in our testing process, and the only easy source of that test data is a production file that has that specific problem	X9 testing tools can be used to automatically create one or more image mismatch sequences within an x9 file.
9) We need to include invalid tiff image conditions in our testing process, and the only easy source of that test data is a series of production x9 files that include those problems	X9 testing tools will allow you to create a wide variety of invalid tiff image conditions. Conditions can be automatically generated that will test all x9.100-181 image validations.
10) We need large x9 files for volume and stress testing, and there is no reasonable source of large test files other than the use of production x9 files	X9 testing tools can be used to automatically create very large x9 cash letter files. A variety of strategies exist including generation from your test cases, the use of randomly created data, or the cloning of a small number of transactions to create very large x9 files.

X9 Testing Challenges

There are several issues that complicate the creation of x9 test files:

Challenge	Why it is Complex
1) Define test cases	Tools must be used to allow test cases to be easily defined, communicated, and maintained.
2) Build and maintain a test case repository	A repository of test cases must be constructed and stored (a complex and time consuming effort).
3) Create x9 header record configurations	Files must be created where the file and cash letter header records match the validation rules that have been built into the capture system. Otherwise, the x9 file will be immediately rejected.
4) Create x9 records and fields	X9 files are structurally complex with various standards that dictate their creation. Records must be created in the appropriate order, with the required fields, justification, and values. Control totals must match the generated records and fields.
5) Create images that match the MICR test cases	The x9 file must have images that logically match the MICR test case. This requirement for a matching image can be system based, where a sophisticated capture application is extracting information from the check image and comparing it to the type 25/31 record. The requirement for a matching image may instead be more visual in nature, where there are benefits of having a matching image.
6) Create x9 files that contain invalid x9 fields per x9 standards	While it is important to create x9 files with valid data, it is equally important to create x9 files that contain x9 fields that are invalid per x9 rules.
7) Create large x9 files for volume and stress testeing	Large x9 files are inherently difficult to create; you can often run out of memory when working with large x9 files.
8) Merge x9 files	It is difficult to retain the header and trailer records from one x9 file while inserting the bundles and check records from a second file into the first. There are situations where combining x9 files can be very helpful, since you can combine multiple test conditions, across various x9 files, into a single file that can then be tested.
9) Create image mismatch conditions, where the image does not match the x9 records	Image mismatch has become a huge problem in the industry that needs to be regression tested whenever application changes are being installed. However, there is no easy way to intentionally create an image mismatch file. Because of this, many organizations will use a production mismatch file for this testing, which goes against FFIEC testing standards.

Challenge	Why it is Complex
10) Create invalid tiff images (which do not conform to the x9.100-181 standard)	There is a substantial list of tiff image requirements per x9 industry standards. Many image applications will crash when presented with invalid tiff images. However, there is no way to purposely create tiff images that violate the x9.100-181 standard.
11) Allow embedded images to be extracted	Tiff images are embedded within each record type 52. Given the variable length nature of an x9 file, it is difficult to isolate and extract the tiff images.
12) Allow embedded images to be manipulated	It is normally difficult to impossible to manipulate an individual image within an x9 file.
13) Comparison of x9 files	X9 files are variable length and contain embedded images, which makes obtaining their individual record data difficult. In addition, x9 files are typically encoded in the EBCDIC character set, which is foreign to many of the data comparison utilities that exist in the server and workstation environments.
14) Create checks to be independently created and captured for remote deposit or branch capture application testing	Once you solve your x9 testing issues, you must still address your remote deposit capture and branch capture testing needs. These applications require paper documents and cannot utilize x9 files that you create for your image enabled applications.

These certainly do represent complex problems associated with x9 file testing. However, the good news is that as the x9 industry has matured, there are automated tools that address these specific needs. If you are struggling with these issues then you should research the tools that are now available from a variety of software vendors that now exist in this space.

Test Cases

Use Cases

A use case scenario describes the usage of an application system by an actor, with the intent of accomplishing a specific goal. The term “actor” represents a user or other object which directly interacts with the application system. In the general sense, actors represent people who interact with applications. In a more specific sense, an actor can be another computer system or an incoming transaction type. Finally, a usage “scenario” is a sequence of steps that describe the interactions between an actor and the application system.

The use case model (UCM) consists of the collection of all actors and all use cases.

Use cases then:

- Provide a methodology to engage users in the requirement gathering and definition process
- Capture the system's functional requirements from a specific actor’s perspective
- Serve as the foundation for developing individual system test cases

Test Cases

Each Use Case serves as the basis for writing one or more test cases. Using the Use Case Model, you can then create a testing process that is both repeatable and measurable, and will ultimately improve the quality of your production application.

Benefits of this Use Case based testing model is as follows:

- Traceability ensures that each use case is mapped to one or more test case(s)
- The resulting testing process is measurable
    - Each Test Case can be executed and then documented as pass / fail
    - Failed Test Cases can be repetitively tested until passed
- The resulting testing process is repeatable
    - Individual tests can be applied to the current test cycle
    - These tests then become part of a test case repository that can be used in future tests
- The Test Case Repository can be updated and improved over time, as:
    - Undocumented requirements are identified and included in the definition
    - Requirement gaps are closed
    - Functional requirements evolve
    - Application maturity increases
- The Test Case Repository can serve as the basis for automated testing and tracking

MICR Test Cases

The following data elements comprise an individual debit or credit MICR item:

- Amount
- ABA number (including optional ABA check digit)

- Account number
- Process Control field
    - Transaction code
    - Check serial number
- Auxiliary OnUs
- Field 4
- EPC

There are several important items that must be considered and accommodated:

- The chosen ABA must typically correspond with the assigned account number
- All MICR line fields must individually and collectively be acceptable by the capture logic
- If captured transactions are being passed to downstream applications:

    - The account must be defined within the test bed
    - The account must have attributes (status, balance, etc) to support the transaction
    - The transaction code (process control) must be appropriate for the account
    - The check serial number (process control or Aux OnUs) must be appropriate for the account

Debit Only versus POD Files

In the check processing environment, work can typically be viewed as “debits only” or “POD”. A POD (proof of deposit) environment is defined to process logical transactions, where one or more credits are offset by one or more debits. In a true POD environment, there are validations that are built into the processing flow which will ensure that the encoded amounts on each item match the assigned processing amount, and that each customer transaction is logically in balance debits to credits.

Your test case repository can be defined with your desired combination of credits and debits, and in the order required by your capture application (eg, credits first).

Summary of Transaction Set Requirements

At a high level, you should consider the follow requirements for your testing tool:

- Ability to use industry standard tools to define your use cases
- Ability to provide full MICR line encoding at the field level
- Ability to control the text that is placed within the drawn images
- Ability to create debit only cash letter files
- Ability to create POD cash letter files
- Ability to support single credit and multi-credit deposits
- Ability to control how items are bundled into cash letters

X9 Data Creation for Unit Testing

Creating an x9 file from Randomly Generated Data

The individual fields that make up a MICR test case can be created using random data generation techniques (eg., the use of a random number generator). This general approach is useful when you want to generate very large amounts of data with a minimum amount of effort. However, the use of random data is generally insufficient and unacceptable when you need targeted data that will specifically exercise your documented test cases.

With that said, random data does not have to be completely "random". You can generate random data where the contents are much more useful than arbitrarily created random numbers.

- ABA numbers can be randomly selected from a list of industry defined ABA numbers. This approach will result in a list of random ABA numbers that will self-check and should be acceptable to your capture application as well as to your downstream applications.

- Account numbers can be generated within defined account number ranges and using the appropriate self check routines. The generated account numbers can then be associated with the appropriate ABA number. This approach will result in account number / ABA pairs that can approximate the information that may be seen in a production transaction processing environment.

- Amounts can be assigned sequentially within a defined range (for example, each amount incremented by one cent). This approach has several advantages. First, it improves traceability by ensuring that each amount is unique. Second, it allows large files to be generated within the aggregate dollar limits required by x9 files.

- Sequence numbers can be sequentially assigned, which improves traceability.

Image Creation

A key factor associated with the automated creation of x9 data files is the generation of check images that logically match the x9 data itself. When images are drawn to match the generated data, the resulting x9 files will have enough validity to be accepted by your x9 image capture applications. Dynamically drawn images can utilize check artwork and fonts to create images that are based on the randomly created data. Specifically:

- A handwriting font can be used to place the random amount into the courtesy amount box
- A handwriting font can be used to place the random amount into the legal amount line as text
- A MICR font can be used to format and draw the OCR MICR line per industry standards

Random Data as the Basis for Unit Testing

Are there benefits from testing with randomly generated x9 data? Absolutely !!

There are various methods to create the new data element that is being assigned:

- Randomly generated
- Assigned randomly from a predefined list
- Sequentially assigned within a defined range
- Calculated based on specific field
- Intelligently assigned based on the content of other fields which are present

Using such techniques allows “random” values to be assigned that are appropriate for the targeted field and that will pass all of the validations (edits) that are defined at the field level.

Although random data is quick and easy to create, it is also simplistic in nature and does not ordinarily represent the more complex (and accurate) test cases that can typically be defined to be appropriate for downstream application testing.

Use Case Definitions

Your use case definitions should ideally be defined using an industry tool (such as MS-Excel) that is widely used within your organization. This will allow you to share and distribute the tool to various users and departments that have x9 testing requirements. This will allow them to define their testing requirements and provide them to a centralized testing group that can organize, facilitate, and execute the test. The following is such an example:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	*				Sample Deposit File											
2	*								Tran		Sequence			Check	Transaction	Image Print
3	*	D/C	AuxOnUs	EPC	ABA	Field4	Account	Serial	Code	Amount	Number	Payee	Memo	Format	Description	Eligible?
4		C			103000648		74345034	9902		0.00	45000004	Test Transaction	ISN: 45000004	business3	:: Teller Test ::	
5		D	188459		2118-7202		34450778			205.18	45000005	Test Transaction	ISN: 45000005	business3	:: POD Test ::	
6		D	4440102		2212-4145		86920667		52	12.00	45000006	Test Transaction	ISN: 45000006	business3	:: Branch Capture Test ::	
7		D			312270010		57736246	7822	52	44.53	44000007	Test Transaction	ISN: 44000007	business3	:: Lockbox Test ::	No
8		D			103102216		12069606	3309		79.08	45000008	Test Transaction	ISN: 45000008	business3	:: ATM Test ::	
9		D			221275533		66802675	3696		115.44	45000009	Test Transaction	ISN: 45000009	business3	:: Teller Test ::	
10		D			55001151		78026113	1601		53.79	45000010	Test Transaction	ISN: 45000010	business3	:: POD Test ::	
11		D			101910617		77523882	9286		27.27	45000011	Test Transaction	ISN: 45000011	business3	:: Branch Capture Test ::	
12		D			291971469		61102578	6977		100.00	45000012	Test Transaction	ISN: 45000012	business3	:: Lockbox Test ::	
13		D			211872027		40259798	9791		157.12	45000013	Test Transaction	ISN: 45000013	business3	:: ATM Test ::	
14		D			11100106		26794823	4217		214.33	45000014	Test Transaction	ISN: 45000014	business3	:: Teller Test ::	
15		D			11301073		54927101	2321		188.15	45000015	Test Transaction	ISN: 45000015	business3	:: POD Test ::	
16		D			26013673		81490429	1499		101.12	45000016	Test Transaction	ISN: 45000016	business3	:: Branch Capture Test ::	
17		D			31100490		91740219	5218		77.15	45000017	Test Transaction	ISN: 45000017	business3	:: Lockbox Test ::	
18		D			31201328		16830072	6128		119.44	44000018	Test Transaction	ISN: 44000018	business3	:: ATM Test ::	No
19		D			41205534		77048126	4419		91.55	45000019	Test Transaction	ISN: 45000019	business3	:: Teller Test ::	
20		D			41203824		29585281	3327		52.18	45000020	Test Transaction	ISN: 45000020	business3	:: POD Test ::	
21		D			42103363		88011891	3350		155.87	45000021	Test Transaction	ISN: 45000021	business3	:: Branch Capture Test ::	
22		D			11075150		51057394	4278		121.00	45000022	Test Transaction	ISN: 45000022	business3	:: Lockbox Test ::	
23		D			101000077		18721601	4546		79.54	45000023	Test Transaction	ISN: 45000023	business3	:: ATM Test ::	
24		D			123103729		61529244	3831		42.11	45000024	Test Transaction	ISN: 45000024	business3	:: Teller Test ::	
25		D			114900685		28794481	3899		98.15	45000025	Test Transaction	ISN: 45000025	business3	:: POD Test ::	
26		D			62103136		14295520	3028		33.56	44000026	Test Transaction	ISN: 44000026	business3	:: Branch Capture Test ::	No
27		D			107089555		58143303	3967		67.88	45000027	Test Transaction	ISN: 45000027	business3	:: Lockbox Test ::	
28		D			75906951		31991822	2851		59.12	45000028	Test Transaction	ISN: 45000028	business3	:: ATM Test ::	
29		D			62202419		55149217	2839		133.59	45000029	Test Transaction	ISN: 45000029	business3	:: Teller Test ::	
30																
31																
32																
33																

However, randomly generated test data can be very appropriate for x9 unit testing. This is especially true if you generate x9 files using a single ABA and with account numbers that are assigned from your account ranges and incorporate your self-check routines.

As an example, you can easily:

- Define one or more ABA numbers to be used for your test cases
- Define your self-check (MOD check) routines
- Generate account number lists for each of the ABA numbers you have defined
- Combine those generated lists into a single master MICR use case list
- Create several x9 files containing 10,000 checks each from your randomized use case list

These x9 files can be used by your Unit Testing phase, which brings this critical and frequently used testing phase into FFIEC compliance.

Sanitized Data Used as the Basis for Unit Testing

An ***existing x9 file*** can be used as the basis to create a new x9 file using data "scrub" or "sanitization" techniques.

Creating a sanitized x9 file is a straight forward process that is performed against one of your existing x9 files. The resulting x9 file can either preserve or mask the original file and cash letter header attributes. You can then decide to retain or scrub as much of the transaction level data as desired to meet your specific testing objectives.

When you sanitize an individual data element, you are replacing its current value with an alternative value that will mask its original content. For example, you can scrub an account number by replacing the value "3112578946" with "123456789". By doing this, the original customer account number has been obliterated and will no longer be assigned to the transaction. Despite this data replacement and confidentiality that has been obtained, much of the intent of the original x9 transaction remains.

There are numerous benefits obtained by sanitizing existing x9 files to obtain new x9 test files:

- You can quickly create x9 files without the effort of defining individual test cases
- You can select which data elements are to be sanitized (masked)
- The resulting transactions will exercise either all or a signification percentage of the functionality of your x9 image based capture application.
- The resulting transactions can be readily accepted by edits and downstream applications
- Confidentially of the original transaction improves as the number of scrubbed fields is increased

Sanitizing just the account number provides a large degree of confidentially for a MICR transaction.

You can build on this logic by also scrubbing the ABA, check serial number, amount, and the tiff image. Once all of these fields are scrubbed as a group, you will have a fully masked transaction which will ensure that all customer confidentiality requirements have been met.

Sanitizing x9 Files by Applying Defined Use Cases

In the real world, MICR line fields cannot be assigned from completely random data. Very specifically,

- The account number is logically dependent on the ABA
- The account number must be constructed per OnUs rules
- The check serial number is logically dependent on the account number
- The auxiliary OnUs field is logically dependent on the account number

By sanitizing these fields as a group (by assigning them from your Use Case file), you can ensure that the newly assigned values will work together to meet your objectives. For example:

- Account numbers are valid for the specified ABA
- Account number lengths are appropriate
- Account number check digits are appropriate
- Check serial numbers can be defined as needed (eg, for controlled disbursements or drafts)

Because of these basic requirements, MICR line fields should be logically sanitized (modified) as a group. This data grouping consists of the following MICR line fields:

- ABA
- Account number
- Check serial number
- Auxiliary OnUs

Benefits of Testing with Sanitized Data

Are there benefits from testing with sanitized x9 data? The answer is an immediate yes !! However, the more complex question is what fields should be replaced by the scrub operation. At a minimum, the following x9 data must have the potential to be sanitized to ensure that customer data is protected:

- ***MICR line fields***
- ***Amounts***
- ***Images***
- File header ABAs
- Endorsement ABAs

You can then additionally consider sequence numbers, user fields, reserved fields, and various other x9 data for sanitization based on the above core requirements being met.

When you sanitize at least the MICR line fields and the images, you will generate a new x9 file with all customer confidential information obliterated.

These x9 files can be used by your Unit Testing phase, which brings this critical and frequently used testing phase into FFIEC compliance.

Summary

Although the x9.37 related standards have been in use for 10+ years, practitioners of these check image exchange specifications must constantly address and adapt to ongoing changes within the industry. This ongoing change includes the need to strengthen our testing procedures to minimize the use of production data within these environments. This need is mandated by internal and external audit requirements and is a necessity to protect the confidentiality of customer data.

Within the problem statement of this white paper, it was stated that a fully functional X9 data generation tool must support the following functions:

- Creation of x9 files (data and images) from use case definitions
- Ability to build a repository of test cases which can be used for regression testing
- Ability to sanitize production files to easily create representative test files
- Ability to create high volume stress files from sanitized production data

Software tools do exist that address the above stated requirements. Use of these types of tools can not only increase your productivity but also ensure that data privacy requirements are met. Moving towards an x9 test case repository will help test teams improve the repeatability measures associated with their applications, improve the reliability of those systems, and move to higher levels within the Capability Maturity Model.

Appendix: X9Assist as a Testing Tool

About X9Ware LLC

X9Ware LLC provides extensive tools for users of the various x9.37 file specifications. Our product line extends from a free x9 viewer to tools that include validate, modify, repair, make, generate, scrub, import, export, and provide a host of other x9 support and reporting functions. We have price competitive licensing options to meet the needs of any size organization. Our goal is to offer what we believe are some of the best tools in the industry and at the lowest possible cost.

X9Assist is our flagship product which incorporates a series of tools to support your x9 production support, testing, and development requirements. Specific examples are:

Tool	Usage
1) Make	Creates a new x9 file from a use case file which can be defined in either Excel or CSV format. A variety of fields can be provided via the use case file. Fields that are not present in the use case file can be assigned constant or random data based on your specific requirements.
2) Generate	Creates a new x9 file from a CSV file which can be obtained from various sources. The most typical usage of Generate is as the second step of the Make process (where the output from Make is the input to Generate).
3) Scrub	Obliterates fields within an existing x9 file by replacing values in selected fields with either random or use case data. The x9 data elements that can be altered include MICR line fields, amounts, item sequence numbers, endorsements, and file header records. Scrub also redraws images associated with an altered check after the new MICR fields are assigned, thus allowing the x9 data and the image to remain synchronized.
4) Clone	Clones bundles within existing cash letters to create large x9 files for performance and stress test purposes.
5) Merge	Merges cash letters from multiple independent files into a single, consolidated x9 file that can be used for testing purposes.

For additional information on the capabilities of X9Assist, please contact lowell.huff@x9ware.com, or visit our website at www.x9ware.com.