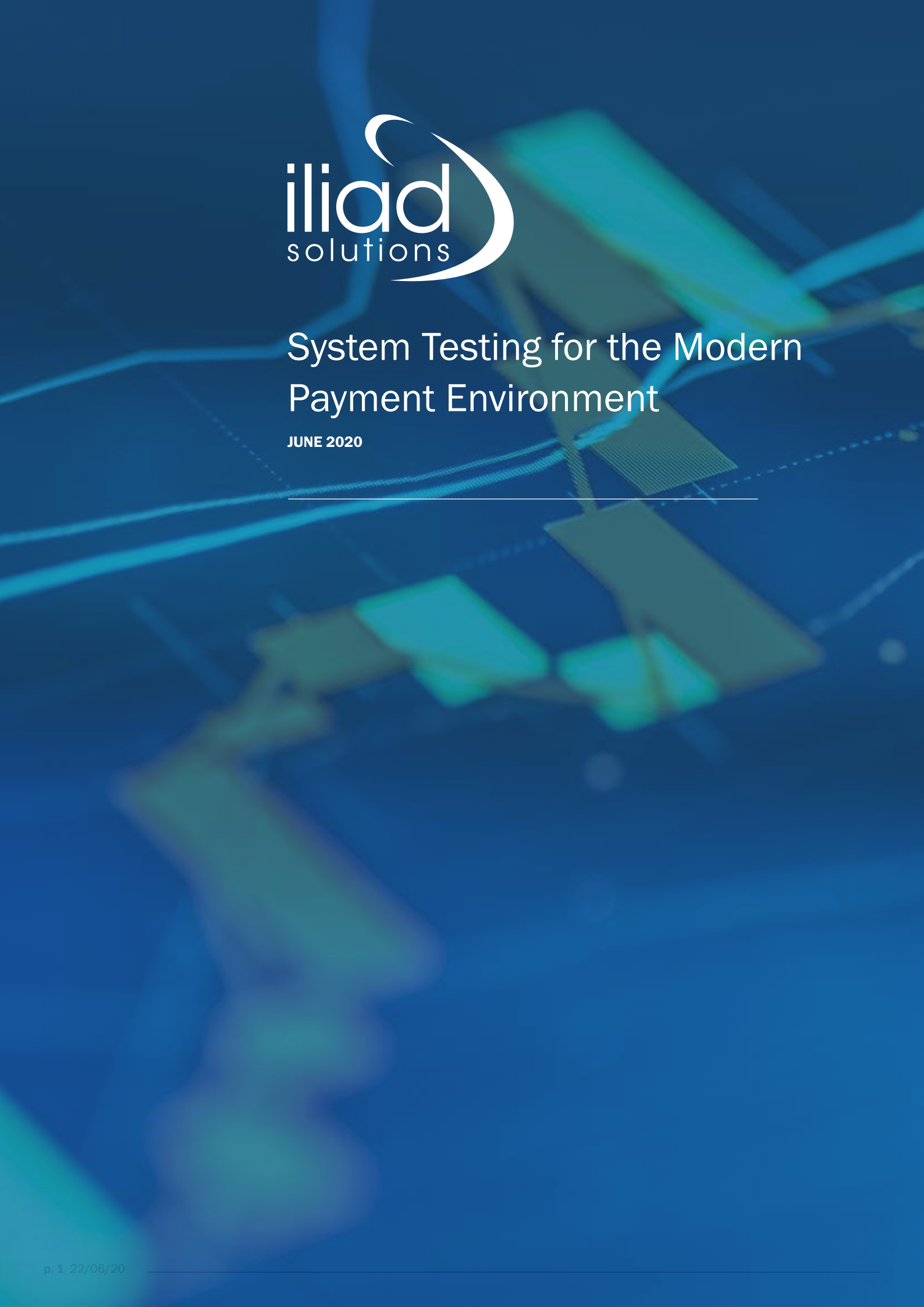

System Testing for the Modern Payment Environment

JUNE 2020

Introduction

Every organization in the payments industry has been affected by unprecedented levels of change. New regulations, new competitors, new technologies and the expectations of a new generation of tech-savvy customers are making more demands on technology teams than ever before.

To meet these new demands, banks have embarked upon a period of significant development of the IT infrastructure that supports their payments business. As well as the increasing complexity of their payments systems, project, business and technology teams face a considerable workload. However, while banks are busy deploying the latest technology and working with imaginative new suppliers, the fundamentals of their testing frameworks and methods are not advancing at the same pace.

This paper explores the current methodology and the reasons why it is no longer sustainable, and maps out the approach to testing that organizations should be taking in the future. By adopting a different approach, organizations can create more opportunities to successfully innovate and reduce risk in a highly competitive environment.

Automated Payment Systems

The four party model shown in Figure 1 illustrates the nature of an Automated Payments System. It is made up of four virtual or physical entities (merchant, acquirer, network and issuer) that communicate using a common protocol based on ISO-standard 8583. There may be different versions of ISO 8583 used for authorization and clearing messages within the network, but the model assumes a common language between the components.

FIGURE 1. THE BASIC FOUR PARTY PAYMENTS MODEL

Testing the Four Party Model

Testing of the four party model is based on the use of a “simulator” system to substitute for one or more of its components. A simulator may generate messages as an acquirer or merchant system, respond to messages as an issuer or even “pass through” messages as a network. Simulator testing of this type has the following characteristics:

- The central component of this testing structure is a standalone “simulator” program, typically executed on separate personal workstations or on a centralized mainframe system
- A limited number of pre-defined tests, depending on the type of test (unit, system, integration, acceptance) and the model being simulated (merchant, acquirer, network, issuer)
- A limited number of pre-defined accounts, banking institutions, merchants, etc. to be used in conjunction with specific tests
- Pre-defined information (keys, connections, etc.) to support the accounts and institutions used by the tests
- All test and other information conforming to a single agreed language or protocol
- All test and other information added manually or by internal custom program that builds up over time
- Changes to the simulator hardware required for changes in protocol and other major operations program

Testing the Four Party Model

Limitations of Traditional Testing Structure

The characteristics of the traditional testing structure constrain the flexibility increasingly required in testing Automated Payment Systems:

1. Standalone PC programs do not integrate easily with external test management programs like QC
2. Manual input of tests and data does not support the amount and variety of test data needed in a multi-protocol environment or for large scale stress and variance testing
3. Substantial changes are accomplished via changes to internal programming
4. External interfaces require external utility programming

Increasing Complexity

Over time, automated payment systems have become more complicated. Different protocols and processing system features have been added, using different messaging protocols. This leads to additional testing demands that have normally been addressed ad hoc. The following case study illustrates a single unitary change that increases the complexity of testing, and may be used as a basis for understanding how the increasing system complexities have an exponential effect on the testing requirements.

Case Study: Acquirer using a New Network Protocol (option 1)

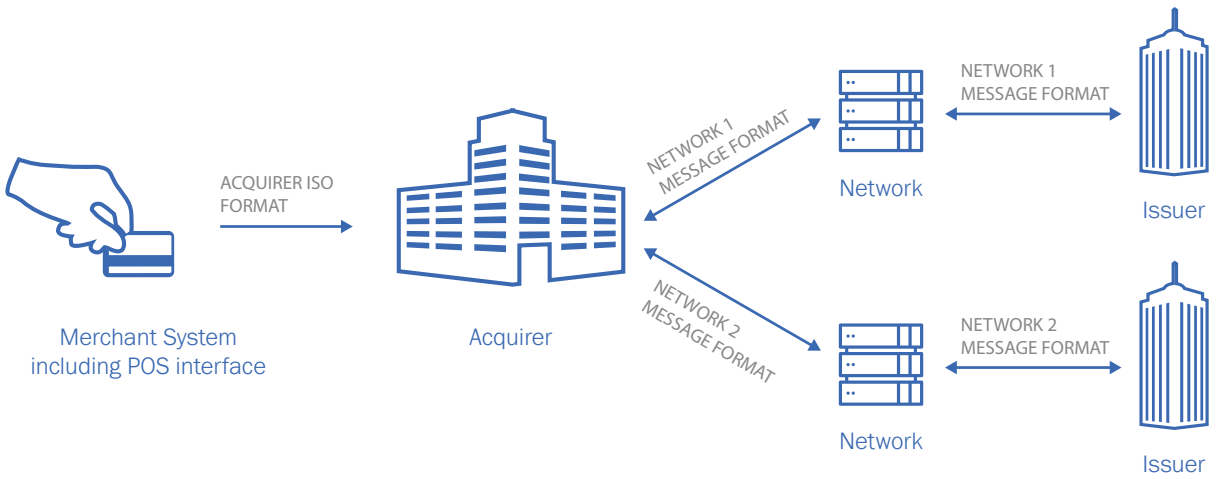

FIGURE 2. THE FOUR PARTY PAYMENTS MODEL WITH TWO PROCESSING NETWORKS

In this example, an acquirer will begin sending transactions to a new network that supports a different protocol from the existing network. Unfortunately, their current testing simulator supports only the original network's protocol. However, the team developing the new protocol interface has found a test simulator that supports the new protocol and has obtained a license for it. This is a different simulator than the one currently in use. Simply using the new simulator is sufficient for unit and stand-alone system testing. However, when integration and acceptance testing of the overall network system is conducted, things are twice as hard

because everything necessary to conduct the test (test creation, data loading, test execution and result analysis) must be done separately on both simulators. And, while the developers are familiar with the new simulator, the integration and acceptance testing teams are not, and must be trained in its use. The result is a doubling of the effort required for integration and acceptance testing, plus the added costs of acquiring and maintaining expertise in the two simulators.

Case Study: Implementing a Rewards (option 2)

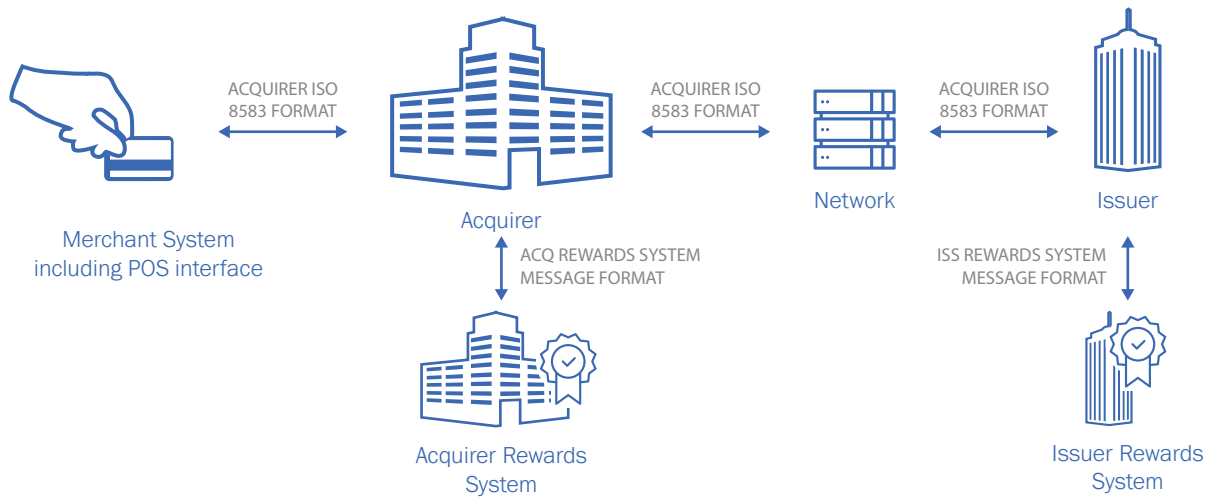

FIGURE 3. THE FOUR PARTY PAYMENTS MODEL WITH ADDITION OF REWARDS SYSTEMS

In this example, one of the entities in the four party model will implement a Rewards system. They will purchase an existing state-of-the-art system that enables the Rewards system to do what they want it to.

However, the Rewards system operates using an XML interface that is not supported by the network testing simulator. Manual XML commands can be generated for unit and system testing but they cannot be supported by the simulator responsible for integration and acceptance testing. As a result, real integrated testing is impossible and can only be mimicked by manually generating XML calls based on the assumed output of the transaction tests.

This manual effort is time consuming and prone to error due to the manual interpretation of the intermediate output.

Testing the Four Party Model

Case Study: Tokenization (option 3)

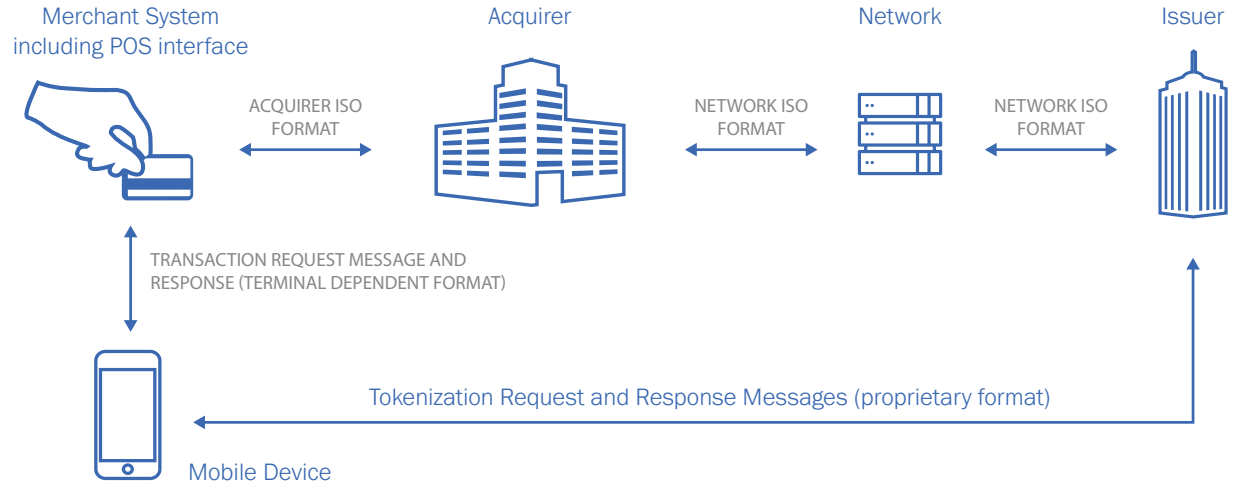

FIGURE 4. THE FOUR PARTY PAYMENTS MODEL WITH ACCOUNT NUMBER TOKENIZATION

The introduction of tokens for secure purchases adds another level of complexity to a basic transaction. These transactions can be generated by the integration of mobile phones, computing systems or other smart devices to initiate the transaction request.

Smart devices communicate via unique secure protocols. Unit and system testing typically use manual tools like SoapUI to respond to tokenization requests. These limit the range of tests that can be carried out and require manual updates for changes in the tokenization protocol. Integration and acceptance testing normally use actual smart devices to initiate tests.

The price of procuring the latest mobile phone technology, and the maintenance of firmware updates, adds to the cost, and each test requires manual intervention to initiate. What is needed is a way to automate testing of the network systems using tokenized transactions.

Further Complications - Fraud Scoring Systems

Fraud scoring systems (also known as Risk Management systems) are used by many networks and card issuers to identify transactions more likely to be fraudulent. These typically operate by sending a subset of data elements from each transaction to a separate system which (a) keeps a record of transactions on a specific account and (b) “score” the next transaction with a higher score to indicate the likelihood of fraud. Some Fraud Scoring systems supply scores based on the current transaction or provide a score based on previous transactions in the account. This score is used as part of the approval process for the transaction.

Fraud Scoring systems may use the entire financial request message or a subset of it. They may also require additional data from the approval system. In many of these cases, they have a different message format from the request and response messages. A Fraud Scoring system would fit into the four party model in a way similar to the standalone rewards system, that is, it will be an auxiliary system called by either the network system or the issuer’s processing system.

Testing the Four Party Model

The Impact on Traditional Testing Structure

There are major limitations to the traditional four party model testing structure. These necessitate manual intervention to integrate new capabilities into integration and acceptance testing. While this can be sustained through one or two rounds of additional complexity, the personnel costs, complication of effort, and resulting error rates become very costly over time.

An example is provided in the figure below which combines elements of each of the characteristics described. While complex, it is not an unrealistic representation of the total flow of a financial request message and an associated response.

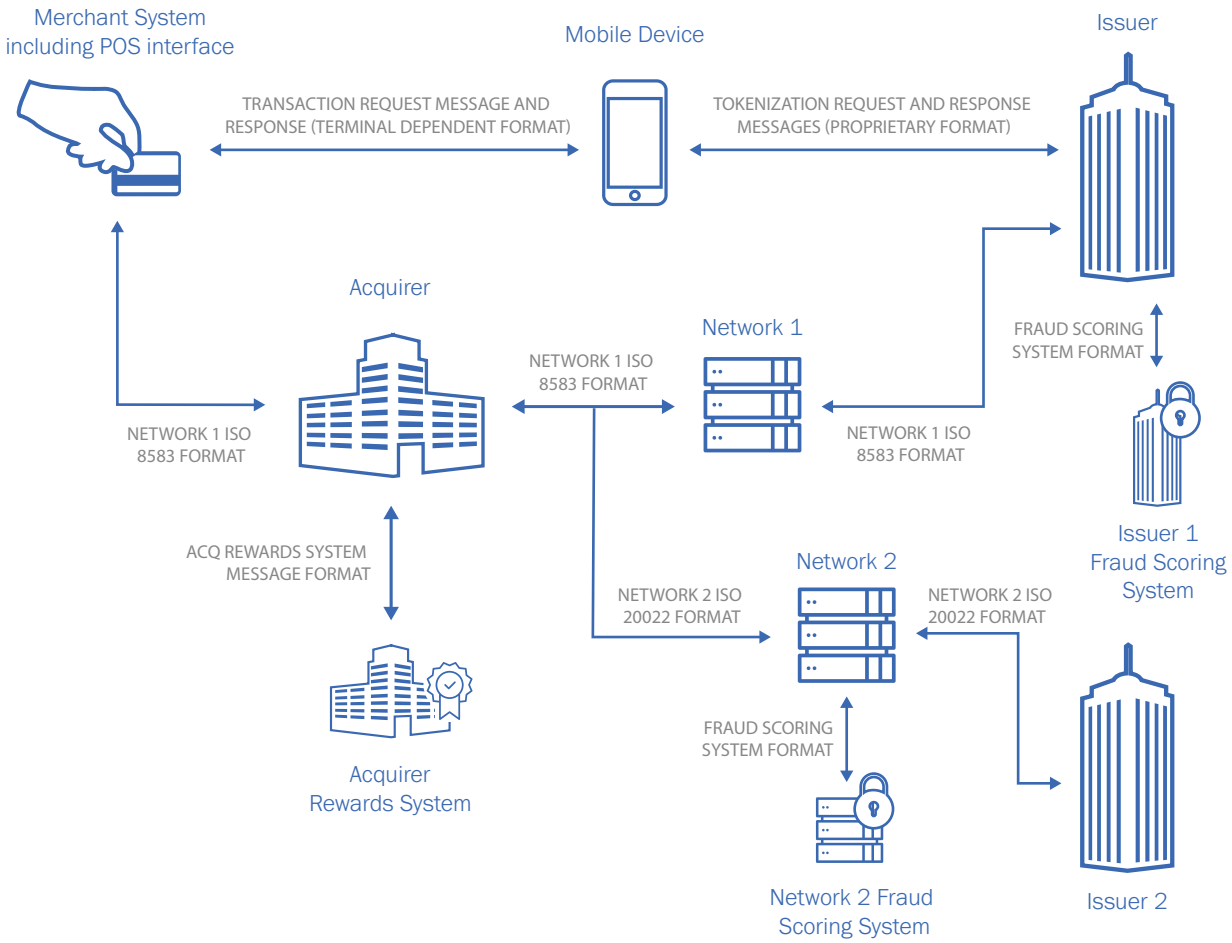

FIGURE 5. FOUR PARTY PAYMENTS MODEL WITH MULTIPLE AUXILIARY SYSTEMS

The Solution

What is needed is a tool that bypasses the limitations of traditional four party model testing and provides flexibility for current and future testing needs.

API-based Software Structure

Iliad's t3:Switch is a multi-level software system based on Restful API calls. While a User Interface similar to many existing testing simulators is provided, customized interfaces to and from external systems can be developed by calling specific functions in the t3 Restful API.
Among other capabilities, this supports integration of t3:Switch into existing test management systems, like QC. In addition, the t3:Switch API can be used by scripting languages like Jenkins or JSON to build programs to automate:

- test execution
- data import and export
- test result export and analysis

Integrated Real-time Payments Testing Platform

Protocol Independence
t3:Switch is protocol independent. In other words, the testing platform software does not depend on a single messaging protocol but is instead based on the internal definition of the protocol, defined in a framework read by the platform. This allows not only a variety of ISO 8583-based protocols to be used by the same testing effort but supports ISO 20022, XML, HTML and other language protocols as the basis for testing.

Protocol Compatibility

Each defined test case in t3:Switch is made of a series of "legs" that represent a system request or system response within the test. The simplest tests contain two legs (a request and a response) but many more legs can be defined. The different legs within a test can also use different protocols to represent different systems participating in the test. Data can be passed from one test leg to another, supporting consistent data transfer specific to the test. An example is a tokenized transaction request, illustrated in the example below.

In actual testing, only some of the legs defined in the diagram would be used. The legs that represent the "system under test" would be omitted and the message would be routed to the actual system under test. However, using this framework, any one or more of the systems involved in the transaction can be tested:

- mobile device
- merchant system
- merchant POS device
- acquirer system
- network
- issuer system
- token system (vault)

The Solution

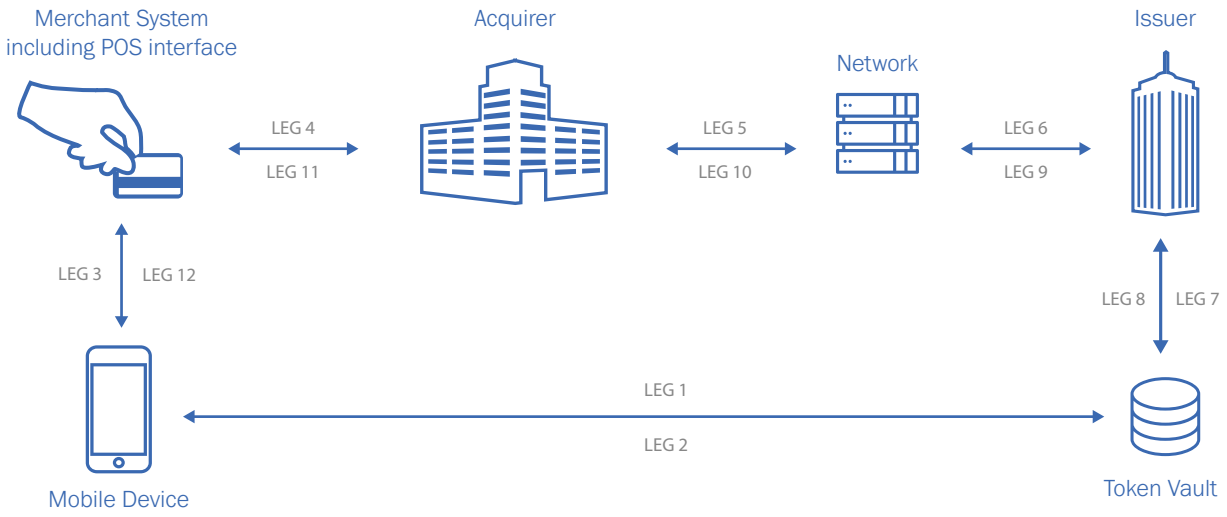

FIGURE 6. TOKEN TRANSACTION DATA FLOW

In this example,

Leg 1 represents a tokenization request coming from a smart device in the protocol used by the device

Leg 2 is a response to the smart device in the protocol used by the device and containing the data from the token vault

Leg 3 is a transaction request generated using the smart device data including a token from leg 2, using POS/mobile device dependent protocol

Leg 4 is the transaction request generated by the merchant system to the acquirer in the acquirer-defined protocol

Leg 5 is the transaction request from the acquirer to the network in the network's defined protocol

Leg 6 is the transaction request from the network to the issuer in the network's defined protocol

Leg 7 is the request to the token vault (in the protocol used by the token vault) to translate the token from the smart device contained in leg 3

Leg 8 is the response from the token vault (in the protocol used by the token vault) with the account associated with the token from the smart device contained in leg 3

Leg 9 is the issuer response to the leg 6 network message using the network protocol

Leg 10 is the network response to the leg 5 acquirer message using the network protocol

Leg 11 is the acquirer response to the leg 4 merchant message using the acquirer protocol

Leg 12 is the response to the mobile device containing the results of the request, using the data generated in previous legs and formatted in the protocol used by the device

The Solution

Certification Platform

To support managed testing for software releases among different internal groups or external customers t3:Switch provides a Certification platform. The Certification platform provides a user interface that allows users to alter a subset of the test data as needed, but requires a pre-defined set of tests that must be successfully executed. While the tester is limited to completing a specific set of tests and operations, the testing manager may manipulate all the test data via the standard User Interface or other API-based operations.

Automation

While greater automation in testing can decrease costs, increase efficiency and, critically, reduce risk, there is no point in organizations simply automating their existing bad testing processes. If you automate a bad testing process, no matter how well, it just becomes a bad automated testing process. Further, it is essential that testing reflects the real world, and the full end-to-end process with all its supporting interfaces and system foibles.

In addition to the protocol and test definition flexibility described in the section above, t3:Switch provides a number of methods of automating different functions of the test process to allow repeatable testing to be conducted easily and efficiently.

Test Execution Automation

The t3:Switch allows a set of tests to be grouped and executed together within the standard user interface, also referred to as the “Web Browser”. These tests can be repeated if desired, and can be combined with other Web Browser capabilities to conduct stress and performance tests effectively and efficiently.

In addition, within the Web Browser , test data sets can be combined and selected randomly by the platform to be used by repeating groups of tests. Data that can be varied includes card numbers, merchants, acquirers and much more. This allows tests, including stress and performance tests, to be repeated with random data variables more realistically than just continually repeating them with the same data.

Beyond the capabilities of the Web Browser, the underlying t3:Switch Restful API allows customized test groups with varying test data to be executed in whatever order, data, etc. the user specifies. The testing can be controlled by using a scripting language or be integrated into a broader test management tool. Using the underlying API allows flexible execution and evaluation of tests without human intervention

The Solution

Data Transfer Automation

Import/Export Function

While t3 systems on the same hardware platform share data, data security may require some tests to be conducted using t3:Switch installations on different platforms (t3:Switch instances) behind separate firewalls. The Web Browser supports the import and export of most of the data used by t3:Switch. This can be used for data transfer to other software systems, but it is most effective for moving data from one t3 instance to another. The ability to export or import data between systems within different firewalls keeps test data consistent between the different t3:Switch instances.

For example, a new set of tests can be developed in the integration testing instance of t3:Switch. These are contained within a set of testing environment firewalls, but need to be moved to the acceptance testing instance within the production firewalls. The tests can be exported from the integration instance into a file that can be imported into the acceptance instance without modifying the file, or to either of t3:Switch's different instances.

API Data Transfer

The API functions provided with t3:Switch support customized movement of data to and from the user's systems into and out of t3. For example, the user may maintain a database of accounts that are separate from the production (or test) execution systems.

During normal processing, the production execution system reads the data from the database programmatically. t3:Switch's API allows that ability to be simulated within the testing platform either on demand or through batch data transfers.

Database Access

Finally, the database used by t3:Switch exists separately from the platform itself and can be manipulated directly using SQL or other database interface languages. This is particularly effective for large-scale database changes or searching for particular combinations of data. It can also be used for data integrity examinations and external data audits.

Result Analysis Automation

The t3:Switch Web Browser provides the ability to compare the results of two tests without a field by field analysis. Two tests are selected and the results are compared with one button click. This can be effective in analyzing test failures against similar tests that succeed.

Results is also available in the API interface or through database query commands. This supports automated comparison of tests from one run of test execution to another. Ultimately, this can be extending to support large-scale regression testing by an automatic comparison of tests from one system release to the previous one.

Iliad Solutions

For over 25 years, Iliad Solutions has been at the forefront of building, implementing and supporting major payment solutions. Our experience has led us to develop the most comprehensive and resilient test solutions available in the world today, and our global customer base trusts us to take the risk out of payment testing.

To find out more about how we can help you visit our website at www.iliad-solutions.com