EXACTPRO TEST AUTOMATION SOLUTION FOR DLT-BASED POST-TRADE INFRASTRUCTURES

White paper

exactpro
EXITUS ACTA PROBAT

Build Software to Test Software

1. Introduction

As an R3 technology partner and a member of Hyperledger, Exactpro is familiar with the software testing challenges of a DLT-based post-trade infrastructure. To meet the needs of a growing number of companies widely using blockchain technologies to automate their business services, Exactpro specialists have performed a DLT post-trade case study based on the open source Corda and Hyperledger technology.

HYPERLEDGER
GENERAL MEMBER

LINUX FOUNDATION
SILVER MEMBER

r3.

FIX TRADING COMMUNITY™
INDUSTRY-DRIVEN • INDEPENDENT • NEUTRAL
PREMIER GLOBAL MEMBER

The subject of the case study is a comprehensive test automation framework adopting End-to-End testing approaches from multiple Exactpro post-trade projects conducted with the help of the company's core testing tool ClearTH. The ClearTH Test Automation Framework helps to test blockchain and distributed ledger (DLT) systems and ensure that they meet the functional and non-functional requirements applicable to every business service in the post-trade infrastructure. The framework is essentially a test harness product providing an innovative solution to be used at the software quality assurance (QA) phases of the Software Delivery Life Cycle (SDLC) for blockchain technology and other DLT systems.

2. Post-trade Infrastructure Challenges

When it comes to testing, some aspects inherent to all post-trade infrastructures pose certain challenges. These are:

1) A high number of infrastructure components
2) Upstream and downstream system dependency
3) Participant structure
4) Trade/Xfer/Position/Account lifecycle
5) Abundance of asset classes
6) Risk calculation
7) Access via a variety of API endpoints (FIX, SWIFT, Custom API)

3. DLT-specific Challenges

Additionally, DLT-based post-trade platforms are characterized by even more complexity brought on by the DLT technology itself. Their list includes:

1) Smart contracts validation
2) Operation outcome only being visible on the ledger (testing software access to the ledger may be limited)
3) Specific development cycle: agile, CI/CD
4) Performance, SLAs
5) Regulatory compliance
6) Client-driven flexibility
7) New business model changes
8) Segregated service delivery

4. A ClearTH Case Study for a DLT-based System — a Detailed Look

Exactpro specialists conducted a case study dedicated to post-trade DLT systems, recreating, in a simplified manner, the processes that take place in post-trade. The case study was performed on the open source blockchain-inspired technology by Corda. To prove that the ClearTH test automation harness comprehensively covers the DLT needs, a suite of functional, non-functional and integration tests were developed for a simple Post-trade DLT-based application. The ClearTH test Automation Framework allows to automate the regression test library and execute complex multi-day test scenarios for such platforms in one go.

All nodes in the network can have permission to create new transactions. Depending on the application, one end user may be one node or may connect to one server node using the web application. In both cases, the end user is free to create new transactions. After a transaction is created, it has to go through the validation and the confirmation stages. What ClearTH does here is it emulates the nodes in the DLT-based systems and verifies the responses.

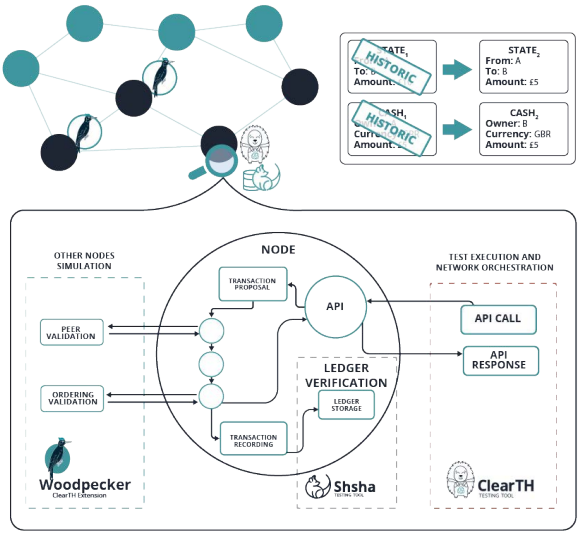

Figure 1.
ClearTH test harness customised for Corda

5. The Testing Scenario: Post-trade Business Flow Steps

In every system, the business flow has its own cycle. In the scenario under review, the initialisation begins at the Start Of Day (SOD), where the system triggers the initial position creation. At this step, in accordance with the ledger, the main functionality is being shared across the relevant parties. The shared positions in the vaults on all 3 nodes (i.e. parties) are at the SOD-state (in our case, all 3 of them equal '0'). The next step of the flow is a new trade creation and its distribution to the relevant nodes within the ledger. In real life, it's mainly market operators that initiate a trade. In other words, the application that they use is a source for the trade before it comes to the post-trade systems. This means that the developed DLT-based application, on par with many other non-DLT-based post-trade systems, has a 'middle layer' that can be called an ETL (Extract, Transform, Load) layer.

So, there is a FIX TCR (Trade Capture Report) message generated and transformed to the Bank A node which, in turn, creates a trade state from this FIX TCR message and shares it with a counterparty — Bank B and the Central CounterParty (CCP). Both nodes, Bank B and the CCP (if they deem this trade relevant), sign this trade (let's id it as TradeID#1) within the ledger. The signed trade triggers another algorithm — 'a positions update' — on the CCP node. In this case study, the positions update business flow has the logic of updating the states in accordance with deal details, such as the side, the quantity, the currency, the asset class, the participants, etc. In the end, the updated position states for both participants, Bank A and Bank B, are distributed across the ledger to the relevant parties. In order to complete the logic flow in accordance with the distributed ledger technology approach, both banks are expected to sign the positions after their internal comparison.

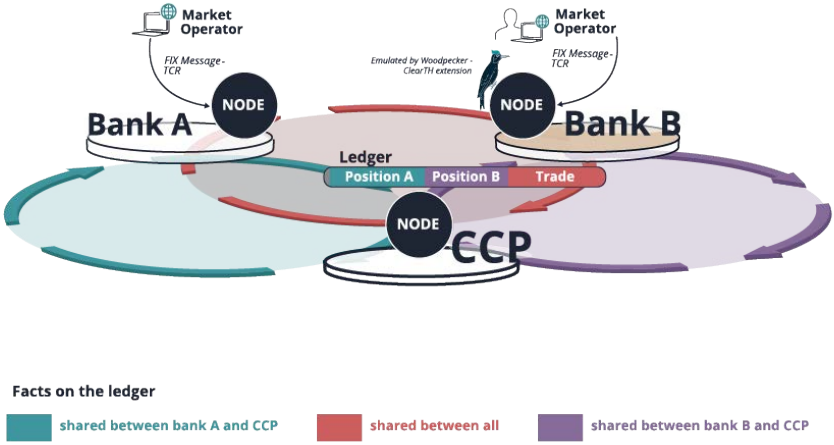

Figure 2. DLT-based Post-trade Business Flow, the main steps

1st step	2nd step	3rd step
Initialisation of positions	Trade creation and participant notification	Positions updates

The post-trade ledger business flow scheme is illustrated in Fig. 2, highlighting the nodes, the flows and the ledger facts. The simple post-trade DLT-based system is now available for further study. Now, let's consider the steps, crucial for validating this functionality, and look at how the ClearTH Test Automation Framework fits in with the QA phase of the platform development.

6. ClearTH Test Automation Framework — E2E Approach for Verification through the Post-trade Business Flow

Before deep diving into the verification details, our DLT-based platform under test should be analysed from the End-2-End (E2E) perspective to make sure it matches the level of quality expected by the end-clients as well as for the integration with surrounding applications or other DLT-based systems. This means that, wherever possible and appropriate, the business flow functionality should be validated as a whole. However, another question arises here: what can be done about the end-points that the emulated platform has no influence on? There are several solutions: the first and most preferable one is to use as many real test integration systems as are available to be able to tackle the unexpected integration issues early, while the second one is to have simulators in place as part of the test framework that meets the main functional and non-functional requirements.

For instance, in the case study described above, the market operator can be emulated by the ClearTH Test Automation framework as well as the relevant test schedules on the counterparty node, Bank B (i.e. signing or rejecting states of the trades or positions). In some cases, the main scheduler of the post-trade systems is a separate application built on top of the system and designed to match the exact timings and sequence required for the specifics of the post-trade platforms: settlement and clearing, margin runs, positions updates, regulatory and internal reporting, etc. This application can also be emulated by the ClearTH test automation framework plugins. However, the triggering events can also be done within this scheduler application by another relevant controller in the test framework.

Test automation in the ClearTH Framework relies on four concepts: Scheduler, Global Step, Matrix and Action.

Scheduler — an entity that contains configuration for the ClearTH test automation framework. It includes Matrices (Scripts) and a Schedule — an ordered set of Global steps referred to in the Matrices. The Schedule is intended to correspond to the steps of the business flow, day (or many business days) in the post-trade DLT-based system under test. ClearTH also allows multiple users to test the system concurrently.

Global step — a part of a Schedule. It identifies a group of Actions to be executed. Any Global Steps can be defined in the Scheduler, but they should also be specified in the Matrix being run.

Matrix — a test script that reflects a test scenario to be automated. It is written in the Matrix language and consists of Actions.

Action — a simple action such as sending one message, comparing the expected and the actual data objects, triggering the system event, emulating the system under integration, etc. Each Action belongs to one Global step, has an ID (it is unique within the Matrix), a name (which shows what the Action is expected to be performed) and a set of parameters that affect the Action's behavior. Depending on the scenario, a set of global steps is created and enriched with the required actions. The primary goal of the ClearTH Test Automation Framework is testing business scenarios via verification of bi-directional message flows.

//CreateSODpositions									
#id	#globalstep	#execute	#comment	#action	#ConnectionName	#StateType	#StateCount		
VerifyStartPositions1	CreateSODpositions	TRUE	Check positions state in Bank A	GetAndVerifyStatesCount	banka	PositionState	0		
VerifyStartPositions2	CreateSODpositions	TRUE	Check positions state in Bank B	GetAndVerifyStatesCount	bankb	PositionState	0		
VerifyStartPositions3	CreateSODpositions	TRUE	Check positions state in CCP	GetAndVerifyStatesCount	CCP	PositionState	0		
#id	#globalstep	#execute	#comment	#action	#ConnectionName	#Account	#Currency	#Quantity	#SaveToContext
CreatePositionA	CreateSODpositions	TRUE	CCP creates SOD position for Bank A	CreatePosition	CCP	@{StaticBankA.PartyID}	@{StaticBankA.Currency}	@{StaticBankA.StartAmount}	y
CreatePositionB	CreateSODpositions	TRUE	CCP creates SOD position for Bank B	CreatePosition	CCP	@{StaticBankB.PartyID}	@{StaticBankB.Currency}	@{StaticBankB.StartAmount}	y
#id	#globalstep	#execute	#action	#StateID	#StateType	#ConnectionName	#amount	#account	
VerifyPositionsA1	CreateSODpositions	TRUE	VerifyVaultState	CreatePositionA	PositionState	banka	@{getAmount{StaticBankA.StartAmount, StaticBankA.Currency}}	@{StaticBankA.FullName}	
VerifyPositionsB1	CreateSODpositions	TRUE	VerifyVaultState	CreatePositionB	PositionState	bankb	@{getAmount{StaticBankB.StartAmount, StaticBankB.Currency}}	@{StaticBankB.FullName}	
VerifyPositionsCcpA1	CreateSODpositions	TRUE	VerifyVaultState	CreatePositionA	PositionState	CCP	@{getAmount{StaticBankA.StartAmount, StaticBankA.Currency}}	@{StaticBankA.FullName}	
VerifyPositionsCcpB1	CreateSODpositions	TRUE	VerifyVaultState	CreatePositionB	PositionState	CCP	@{getAmount{StaticBankB.StartAmount, StaticBankB.Currency}}	@{StaticBankB.FullName}	

Figure 3. ClearTH matrix structure

ClearTH Test Automation Framework produces a thorough report about the testing process, detailing the 'passed' and 'failed' execution statuses for all the global steps and actions. If all the steps are highlighted in green, the test is passed. If an outbound message has been found, the comparison table will be generated showing the passed/failed results for the tags of the message being checked.

6.1. Global Steps

The detailed test scenario description below and Fig. 4 demonstrate the target steps of the scheduler, the validation actions with emulation of the predefined end-point conditions which are also controlled by the framework for the proper coverage and stable results. Based on the case study business flow logic, it is proposed that the validation be divided into 3 global steps in the test schedule.

The **1st Global Step** is SOD positions creation. Within this global step, the ClearTH Test Automation framework checks the shared positions in the vaults on all 3 nodes (as noted above, they should equal to '0' positions), then it starts a flow on the CCP node, creating an SOD position for Bank A and checks the shared positions in vaults on all 3 nodes again (i.e., 1 position in BankA and CCP, 0 positions in Bank B). After that, the scheduler plugin in ClearTH starts a CCP flow of the position creation for Bank B. In order to identify the issues in a timely manner, ClearTH repeats the checks of the shared positions in the vaults on all 3 nodes (this time, 1 position in BankA, 1 — in Bank B, 2 — in CCP).

The **2nd Global Step** is trade creation and sharing. The ClearTH Framework checks the shared trades in the vaults of all 3 nodes (in our case, no trades with TradeID#1), then it checks transactions in the vaults on all 3 nodes (no transactions with TradeID#1 as output either). The framework emulates the Market operator and sends a FIX TCR message to Bank A. The Bank A node initializes a flow to create a state from the FIX TCR message and shares it. After that, as proposed, the test framework emulates the signs from Bank B. All of these are being covered by the standard validation for the shared trade in the vaults of all 3 nodes (where 1 trade with TradeID#1 in all 3 vaults) and checks that this transaction is committed and valid in the vaults on all 3 nodes.

The **3rd Global Step** is Positions update. Within the global step, the shared positions in vaults on all 3 nodes are validated (1 position in Bank A vault, 1 — in Bank B, 2 — in CCP). Then, the CCP node updates the position for Bank A which Bank A signs back. The procedure with checking the shared positions in the vaults on all 3 nodes (position update in Bank A and CCP, no updates in Bank B), then the CCP updates the position for Bank B, and Bank B signs (emulated), followed by the — becoming traditional in this testing flow — checks for shared positions in the vaults on all 3 nodes (position update in Bank B and CCP, no updates in Bank A). At each step, the scenario can be multiplied by variations of the events and responses for expanded test coverage.

7. Conclusions

With the developed ClearTH Test Automation Framework, it's possible to provide end-to-end functional and non-functional testing based on the given business requirements and build a comprehensive regression test library for various post-trade DLT-based systems. The described test automation framework has proven itself with several open source DLT-based technologies.