RFR: Test Strategy & Implementation

SUMMIT

SERIES - 1 / ARTICLE - 1

APRIL 27, 2022

Why Test RFRs?

LIBOR (London Interbank Offered Rate) was the key benchmark interest rate used by global lending institutions for decades. However, its role in the 2008 financial crisis led to its downfall. The fall and replacement of LIBOR have made the migration to an invulnerable system imperative in financial institutions across the world. Indeed, the publication of 24 of the 35 LIBOR settings was halted on 22nd January 2022. As LIBOR is phased out, RFR, or Risk-free Rates, are robust alternatives for your institution. Unlike LIBOR, RFRs are backward-looking and do not include a premium for longer-term funding.

It is important to factor in the impact and risks associated with implementing RFRs at your institution. [redacted] works closely with its clients to ensure a seamless migration from LIBOR to RFR. Our RFR testing methodologies are designed to effectively implement RFR while pre-empting and mitigating the pitfalls and risks associated with this complex transition. In this article, we delineate how we successfully implemented RFR at a client institution using our testing strategy to minimize disruption, error, and risk.

Our Testing Strategy

Our methodology studied Risk-Free Rate(RFR)/Overnight Swap Index (OSI) curves to perform an impact analysis on trade/portfolio valuation and accounting. The strategy tested these curves to mitigate the impact of any sizable changes in the variables on downstream feeds or accounting. As a result, we were able to closely determine the maintenance of all variables within the required thresholds and captured any associated financial risks with the implementation.

Our Testing Approach

To ensure this risk mitigation and a comprehensive understanding of any impact, our testing approach was defined on five different levels.

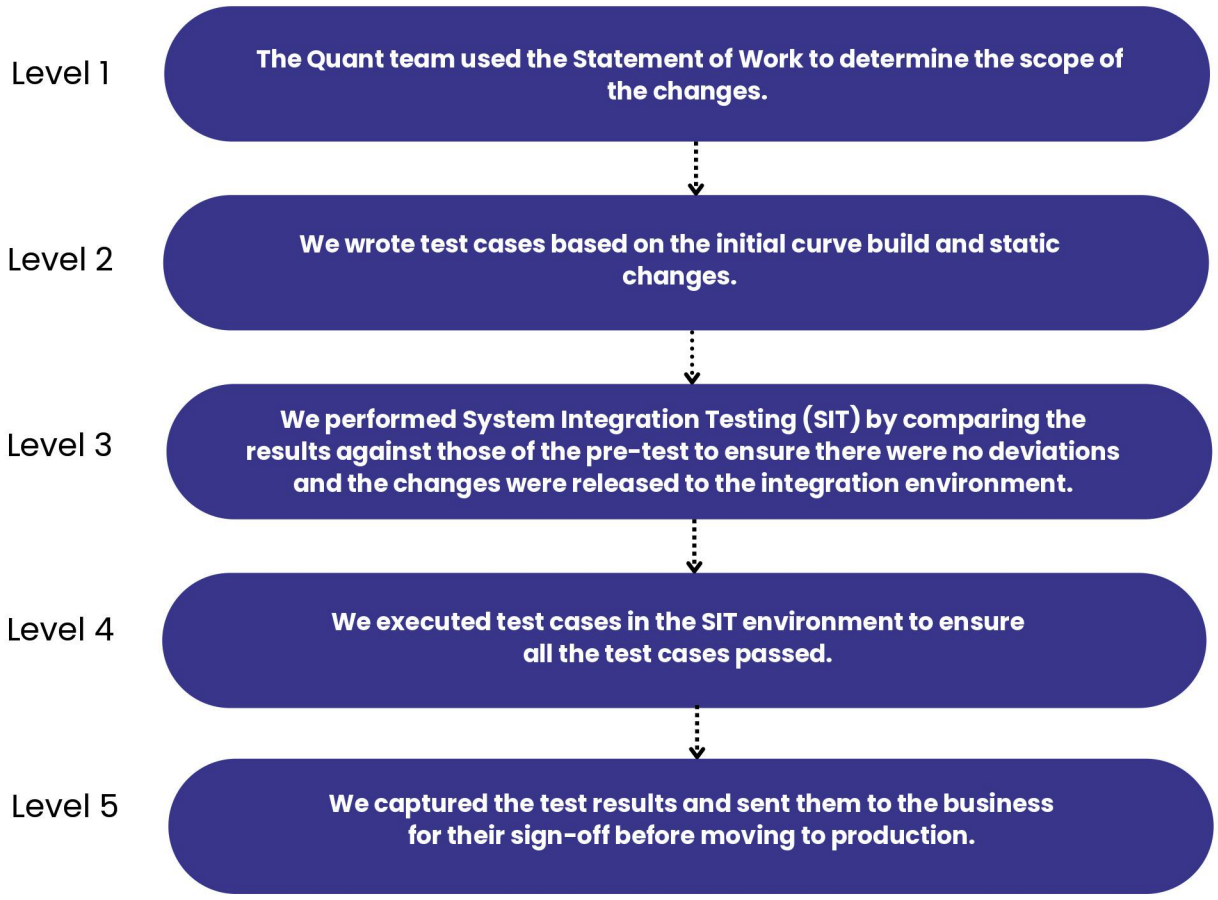

Figure 1: Our 5-level Testing Approach

Our Testing Objectives

- During the build phase, our objective was to validate the RFR/OIS functionality while preserving System Integration Testing and User Acceptance Testing timelines.

- Our quant team oversaw the curve build and the unit testing of the RFR/OIS setup, including curve build changes.

- We performed functional tests to ensure the delivered RFR/OIS functionalities corresponded to those defined in the scope.

- Avoiding impact on the client's business processes during the RFR/OIS implementation was our top priority. Testing was performed incrementally on the existing build to optimize the scope coverage and lower the number of potential defects when entering the SIT/UAT environments. Any potential changes and impacts, including batch processes, were immediately relayed to the client.

The Pre-Testing Phase

- Our first-level testing validated the (i) Initial Build, (ii) Internal Consistency, and (iii) Zero Rate to ensure either a) the zero rate was unaffected, or b) the change was within the permissible threshold outlined in the scope.

- Due to its impact on trade booking, this phase also included the validation of the (iv) Mark to Market (MTM) impact to ensure the pre-implementation and post-implementation values were within the specified thresholds.

- If the difference was above the threshold, we rebuilt the curve to accommodate the difference within the threshold.

The sequence of our pre-testing phase is depicted below:

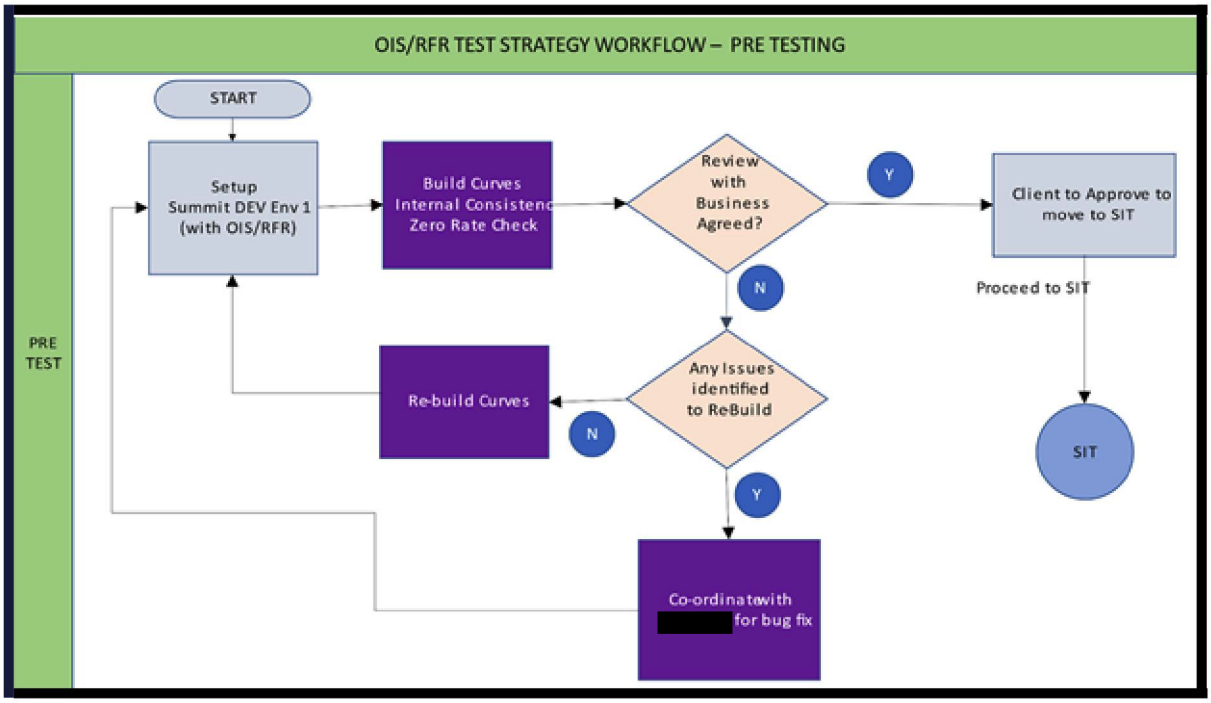

Figure 2: RFR/OIS Test Strategy Pre-Testing Workflow

Regression Testing

We performed regression testing to ensure existing infrastructure and systems at the client institution continued to perform seamlessly after the transition.

- We consolidated all the changes and automated their release into the environment for end-to-end testing. This helped us verify that the curve internal consistency and the MTM difference are the same as determined during the pre-test.

- We then performed downstream system tests. To ensure their prominent impact on feeds and downstream systems, we tested changes in the curve construction and static changes related to the RFR requirements respectively.

- The key test involved comparing the MTM value with the counterparty MTM value to verify that it was matched within the threshold.

- Under this phase, all test cases prepared during the pre-test phase were tested again to ensure that they passed before the production phase.

- All the test results captured during this phase, including the MTM value comparison, were then submitted to the client for their final sign-off.

The workflow of our regression-testing phase is as follows:

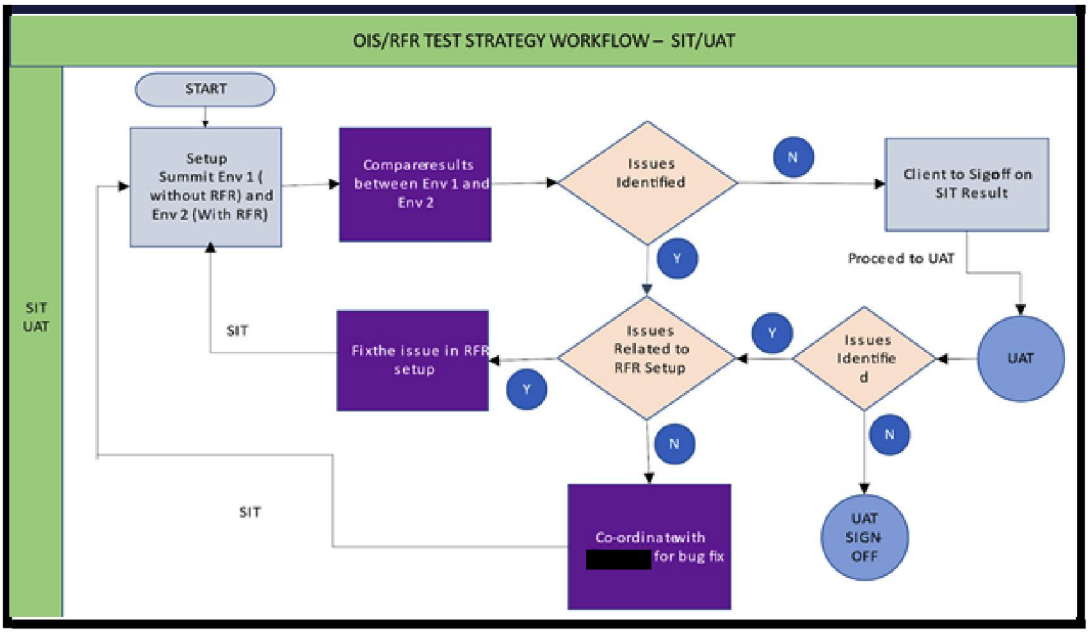

Figure 3: OIS/RFR SIT/UAT Test Strategy Workflow

The Regression Testing Environment

We set up two different environments to illuminate the contrast between an RFR and non-RFR implementation. This also allowed us to easily investigate (i) regression issues and (ii) valuation differences due to the RFR/OIS setup.

Function	Environment 1 With RFR Config	Environment 2 Without RFR Config	Difference
As of Date	Day 0	Day 0	Environment 1 and Environment 2

Table 1: RFR and Non-RFR Testing Environments

Our Roles and Responsibilities

The roles and responsibilities of [redacted]'s project team, our client, and the product are outlined below:

Function	Implementation Team	Client	Product
Test Planning	✓		
Test Reports	✓		
Defect Management	✓		
Refresh Test Environments		✓	
Defect Fixes Related to 6.0			✓
RFR/OIS Configuration	✓		
Smoke/Unit Testing	✓		
Integration Testing Execution	✓		
UAT Execution		✓	
Change Control Management	✓		
Re-testing	✓		
Test Results & Reports	✓		
Test Results & Reports Review	✓	✓	
Test Reports Sign-off		✓	

Table 2: Roles and Responsibilities

Our Testing Environments

We used a number of environments to perform all tests related to the RFR/OIS implementation:

- Environment 1: Functional Setup and Unit Testing
- Environment 2: Integration Testing
- Comparison Testing: Production-like Setup (Environment 1) vs. Integration Testing (Environment 2)
- Client Environment 3: User Acceptance Testing
- Parallel Testing, as applicable

Defect Management

Our defect management system allowed quick and effective responses to any issues that arose during the testing process.

- Identified issues were raised as tickets on the product defect ticketing portal.
- The product team then took the responsibility for the raised ticket and provided a solution.
- After receiving the solution, the project development team then implemented it.
- The functional team then re-tested for the defect to ensure it was fixed.