New Relic Integration - Technical Guide

Test Analytics and Monitoring Platform

INTERNAL DOCUMENTATION

⚠️ **CONFIDENTIAL - INTERNAL USE ONLY**

MQE Unified OAO Test Automation Framework

October 22, 2025

Table of Contents

1. Executive Summary

This internal documentation provides comprehensive technical details about the New Relic integration implementation in the MQE Unified OAO Test Automation Framework. It includes code references, API details, configuration specifics, and implementation guidelines for developers and automation engineers.

> ⚠️ **Confidentiality Notice**
>
> This document contains sensitive technical information including API keys, internal architecture details, and proprietary implementation strategies. It is intended for internal use only and should not be shared outside the organization.

2. Technical Architecture

2.1 System Components

Core Classes

1. TestDataLogger

- **Location:** `com.viacom.unified.utils.dashboard.TestDataLogger`
- **Purpose:** Main entry point for sending data to New Relic
- **Key Methods:**
  - `sendDataToNewRelic(String message)`: Sends JSON data to New Relic Insights API
  - `postTestData(ITestResult result, boolean retryLeft)`: Posts test execution data
  - `postConfigInfo(String configUrl, String version, Map actualConfig)`: Posts app configuration
  - `postCoverageFileLinks(String fileUrl, String status, String testCaseId)`: Posts coverage file links
  - `getTestCaseMappingForCodeCoverage(...)`: Creates test case to code mapping

- `getTotalTestCoverageData(...)`: Generates total coverage data

2. NewRelicData

- **Location:** `com.viacom.unified.service.feed.NewRelicData`
- **Purpose:** Service class for querying New Relic Insights API
- **Key Methods:**
  - `fetchResultsCount()`: Gets count of query results
  - `fetchEventsCount()`: Gets count of events
  - `fetchUniqueTestCaseId()`: Retrieves unique test case IDs
  - `createTestCaseIdAndExecutionTimeMapping()`: Maps test cases to execution times

3. ExportTestRailDataToDashboardToolTest

- **Location:** `com.viacom.unified.tests.sample.ExportTestRailDataToDashboardToolTest`
- **Purpose:** Exports various data to New Relic dashboard
- **Key Test Methods:**
  - `exportTestRailDataToDashboardToolTest()`: Exports TestRail test cases
  - `exportJiraDataToDashboardToolTest()`: Exports Jira bug data
  - `exportTestRailExecutionDataToDashboardToolTest()`: Exports test run data
  - `exportTestCaseMappingDataTest()`: Exports test case to code mappings
  - `extractCoverageDataAppleTest()`: Extracts coverage from Apple platforms
  - `extractCoverageDataAndroidFireTest()`: Extracts coverage from Android platforms
  - `extractCoverageDataRokuTest()`: Extracts coverage from Roku platform

2.2 Data Models

New Relic Models

Location: `com.viacom.unified.model.newrelic`

- `NRResponse`: Main response wrapper
- `Result`: Query result container
- `Event`: Individual event data

- `Metadata` : Query metadata
- `PerformanceStats` : Query performance metrics
- `Content` : Event content data
- `Order` : Result ordering information

Internal Data Models

- `TestExecutionData` : Test execution information
- `TestLabData` : Lab test data
- `CoverageData` : Code coverage data
- `TCItem` : Test case item data

3. Configuration Details

3.1 Properties Configuration

File: `settings.properties`

```
 # New Relic Endpoint
NewRelicEndPoint=https://insights-collector.newrelic.com/v1/accounts/
```

File: `cred.properties`

```
 # New Relic (Base64 encoded)
newRelicKey=TlJJUS0wYnRaUFBaaE1FX0xIeV8yc0VaRTI2LXRSTU50MG5QTw==
```

3.2 API Details

New Relic Insights API:

- **Account ID:** 1519098
- **Insert Endpoint:** `https://insights-collector.newrelic.com/v1/accounts/1519098/events`
- **Query Endpoint:** `https://insights-api.newrelic.com/v1/accounts/1519098/query`
- **Authentication:** API Key based (X-Insert-Key header for insert, X-Query-Key for queries)

3.3 Jenkins Integration

File: `mqe-unified-oao-tests.jenkinsfile`

```
booleanParam name: "SEND_LOGS_TO_DASHBOARD",
             defaultValue: true,
             description: "Enable Sending Logs to Newrelic"
```

4. Event Types and Data Structures

4.1 Test Execution Event

Event Type: `Unified_QE_TestExecution_v1`

```
{
  "eventType": "Unified_QE_TestExecution_v1",
  "timeStamp": "2024-10-22 10:30:45.123",
  "brand": "BET",
  "platform": "Roku",
  "platformType": "TV",
  "appVersion": "5.2.0",
  "locale": "en-US",
  "testResult": "SUCCESS",
  "testName": "validateHomeScreenTest",
  "testCaseId": "123456",
  "failureReason": "",
  "groupType": "Regression,Smoke",
  "feature": "Home Screen",
  "sessionID": "abc-def-123",
  "runId": "jenkins-123",
  "projectName": "Unified OAO",
  "projectId": 167,
  "dataType": "Execution",
  "dataId": "TrdV003",
  "retry": false,
  "sprintName": "Roku_Sprint 2024.10",
  "existingBug": false,
  "issueKey": null,
  "basicDefectType": "To Investigate",
```

```
  "detailedDefectType": "To Investigate",
  "environment": "QA",
  "lineOfBusiness": "Paramount Plus",
  "category": "CTV"
}
```

4.2 Test Case Mapping Event

Event Type: `Unified_QE_TestCaseMapping_v2`

```
{
  "eventType": "Unified_QE_TestCaseMapping_v2",
  "brand": "BET",
  "platform": "Roku",
  "testCaseId": "123456",
  "appVersion": "5.2.0",
  "applicableFiles": "HomeScreen.brs,Navigation.brs,VideoPlayer.brs",
  "executionTime": 45,
  "reportTs": 1698765432000
}
```

4.3 Total Test Coverage Event

Event Type: `Unified_QE_TotalTestCoverage_v1`

```
{
  "eventType": "Unified_QE_TotalTestCoverage_v1",
  "brand": "BET",
  "platform": "Roku",
  "appVersion": "5.2.0",
  "linesCovered": 15420,
  "linesMissed": 8234,
  "branchesCovered": 3245,
  "branchesMissed": 1876,
  "branchCoverage": 63.4,
  "lineCoverage": 65.2,
  "sprintName": "Roku_Sprint 2024.10",
```

```
  "totalFiles": 453
}
```

5. Implementation Details

5.1 Sending Data to New Relic

Code Implementation:

```
 @SneakyThrows
public static void sendDataToNewRelic(String message) {
    HttpPost httpPost = new HttpPost(
        IProps.getConfigSettings().newRelicEndPoint()
    );
    httpPost.setHeader("X-Insert-Key",
        "eb62345fc5fc5b26382f9c527d44e411FFFFNRAL");
    Logger.logMessage("Logging to New Relic: " + message);
    post(message, httpPost);
}

private static void post(String message, HttpPost httpPost) {
    try {
        httpPost.setEntity(new StringEntity(message));
    } catch (UnsupportedEncodingException e) {
        Logger.logException("Failed to set execution data", e);
    }
    try (CloseableHttpClient httpClient =
         HttpClients.createDefault()) {
        CloseableHttpResponse response = httpClient.execute(httpPost)
        int responseStatus = response.getStatusLine().getStatusCode()
        if (responseStatus != 200 && responseStatus != 204) {
            throw new MQETestException(
                "Failed to log response status: " + responseStatus
            );
        }
    } catch (Exception e) {
        Logger.logException("Failed to log in Dashboard " + message, e
```

```
    }
}
```

5.2 Querying Data from New Relic

NewRelicData Service Implementation:

```
public class NewRelicData extends BaseService {

    private static final String NR_ACCOUNT_URL =
        "https://insights-%s.newrelic.com/v1/accounts/1519098/%s";

    private static Map<String, String> requestParameteres =
        new HashMap<>();

    static {
        requestParameteres.put("Content-Type", "application/json");
        requestParameteres.put("X-Query-Key",
            PropertiesReader.getDecodedPropertyValue("newRelicKey"));
    }

    public NewRelicData(String nrQl) {
        super(String.format(NR_ACCOUNT_URL, "api",
            "query?nrql=" + encodeValue(nrQl)), requestParameteres);
    }

    public List<String> fetchUniqueTestCaseId() {
        List<String> testCaseIds = new ArrayList<>();
        int numberOfEvents = fetchResultsCount();
        for (int i = 0; i < numberOfEvents; i++) {
            testCaseIds.add(JsonUtils.fromJson(responseString,
                NRResponse.class).getResults().get(0)
                .getMembers().get(i));
        }
        return testCaseIds;
    }
}
```

5.3 Example NRQL Queries

Fetch Test Cases by File Changes

```
 SELECT uniques(testCaseId)
FROM Unified_QE_TestCaseMapping_v2
WHERE applicableFiles RLIKE '.*(HomeScreen.brs|Player.brs).*'
  AND platform = 'Roku'
  AND brand = 'BET'
  AND reportTs IN (
    SELECT latest(reportTs)
    FROM Unified_QE_TestCaseMapping_v2
    WHERE platform = 'Roku'
      AND brand = 'BET'
    FACET testCaseId
    SINCE 3 month ago
    LIMIT MAX
  )
SINCE 3 month ago
LIMIT MAX
```

Fetch Automated Test Cases

```
 SELECT uniques(testCaseId) as 'Test Cases'
FROM Unified_QE_TestExecution_v1
WHERE platform = 'Roku'
  AND brand = 'BET'
  AND (groupType LIKE '%Regression%'
       OR groupType LIKE '%Smoke%'
       OR groupType LIKE '%Pipeline%')
  AND groupType NOT LIKE '%AdReadiness%'
SINCE 4 month ago
LIMIT MAX
```

6. Code Coverage Integration

6.1 Platform-Specific Coverage Extraction

Android/Fire TV Coverage

Coverage data is extracted from JaCoCo XML reports contained in ZIP files. The extraction process:

1. Downloads ZIP file from JFrog Artifactory
2. Extracts `jacocoTestReport.xml`
3. Parses XML using DocumentBuilder
4. Iterates through method nodes
5. Collects line and branch coverage counters
6. Creates CoverageData objects
7. Aggregates to class and test level

Roku Coverage

Coverage data is in JSON format with method ranges. The extraction process:

1. Parses JSON coverage file
2. Extracts method name and line range
3. Determines if method is covered (boolean flag)
4. Calculates total and covered lines
5. Creates CoverageData objects
6. Aggregates to class and test level

Apple TV/iOS Coverage

Coverage data is in Xcode's JSON format. The extraction process:

1. Parses JSON coverage report
2. Navigates to file nodes
3. Extracts file-level coverage (no method details)
4. Gets line and branch coverage from summary
5. Creates CoverageData objects
6. Aggregates to class and test level

6.2 Shell Script Integration

PostCodeCoverageRunner.sh - Posts coverage file links:

```
#!/bin/bash
```

```
# New Relic Configuration
eventType='Unified_QE_CoverageFileLinks_v1'

# Create JSON payload
jq -n \
  --arg eventType "${eventType}" \
  --arg brand "${brand}" \
  --arg platform "${platform}" \
  --arg appVersion "${appVersion}" \
  --arg buildType "${buildType}" \
  --arg coverageFileLink "${coverageFileLink}" \
  --arg status "${status}" \
  --arg testCaseId "${testCaseId}" \
  '[{
    eventType: $eventType,
    brand: $brand,
    platform: $platform,
    appVersion: $appVersion,
    buildType: $buildType,
    coverageFileLink: $coverageFileLink,
    status: $status,
    testCaseId: $testCaseId
  }]' > coverageFileLink.json

# Post to New Relic
gzip -c coverageFileLink.json | \
  curl --data-binary @- \
  -X POST \
  -H "Content-Type: application/json" \
  -H "Api-Key: eb62345fc5fc5b26382f9c527d44e411FFFFNRAL" \
  -H "Content-Encoding: gzip" \
  https://insights-collector.newrelic.com/v1/accounts/1519098/events
```

7. Dashboard Queries

7.1 Test Pass Rate Over Time

```
 SELECT percentage(count(*), WHERE testResult = 'SUCCESS') as 'Pass Ra
FROM Unified_QE_TestExecution_v1
```

```
WHERE platform = 'Roku' AND brand = 'BET'
SINCE 30 days ago
TIMESERIES
```

7.2 Test Results Distribution

```
SELECT count(*)
FROM Unified_QE_TestExecution_v1
WHERE platform = 'Roku' AND brand = 'BET'
FACET testResult
SINCE 7 days ago
```

7.3 Top Failing Tests

```
SELECT count(*) as 'Failure Count',
       latest(failureReason) as 'Latest Reason'
FROM Unified_QE_TestExecution_v1
WHERE testResult = 'FAILURE'
  AND platform = 'Roku'
  AND brand = 'BET'
FACET testName
SINCE 7 days ago
LIMIT 10
```

7.4 Code Coverage Trends

```
SELECT average(lineCoverage) as 'Line Coverage %',
       average(branchCoverage) as 'Branch Coverage %'
FROM Unified_QE_TotalTestCoverage_v1
WHERE platform = 'Roku' AND brand = 'BET'
SINCE 90 days ago
TIMESERIES AUTO
```

7.5 Flaky Test Detection

```
 SELECT uniqueCount(sessionID) as 'Executions',
       percentage(count(*), WHERE testResult = 'SUCCESS') as 'Success
FROM Unified_QE_TestExecution_v1
WHERE platform = 'Roku' AND brand = 'BET'
FACET testName
HAVING uniqueCount(sessionID) > 5
SINCE 30 days ago
LIMIT 20
```

8. Troubleshooting Guide

8.1 Data Not Appearing in New Relic

Symptoms:

- Dashboard shows no data
- Queries return empty results
- Recent test results not visible

Diagnosis Steps:

1. Check if `SEND_LOGS_TO_DASHBOARD` parameter is true in Jenkins
2. Verify test logs for "Logging to New Relic" messages
3. Check for any exceptions in test logs
4. Validate API key in cred.properties
5. Confirm network connectivity to New Relic endpoints

Solutions:

- Enable dashboard logging in Jenkins parameter
- Fix any authentication issues with API key
- Check firewall rules for New Relic domains
- Verify eventType is correctly set in payload

8.2 Manual Testing with Curl

```
curl -X POST \
  https://insights-collector.newrelic.com/v1/accounts/1519098/events \
  -H 'Content-Type: application/json' \
  -H 'X-Insert-Key: eb62345fc5fc5b26382f9c527d44e411FFFFNRAL' \
  -d '[{
    "eventType": "Unified_QE_TestExecution_v1",
    "testName": "debugTest",
    "platform": "Roku",
    "brand": "BET",
    "testResult": "SUCCESS"
  }]'
```

8.3 Data Validation Query

```
SELECT *
FROM Unified_QE_TestExecution_v1
WHERE testName = 'debugTest'
SINCE 1 hour ago
LIMIT 1
```

9. Security Considerations

> **Security Best Practices**
>
> - **Never commit API keys** to version control
> - Store keys in encrypted properties files
> - Use base64 encoding at minimum
> - Rotate keys periodically
> - Limit key permissions to necessary scopes
> - Monitor key usage for anomalies

9.1 Data Sensitivity

Do NOT send to New Relic:

- Passwords or credentials
- PII (Personally Identifiable Information)
- Credit card or payment information
- Full stack traces with sensitive data
- Internal IP addresses (use placeholders)
- Proprietary business logic

Safe to send:

- Test names and IDs
- Pass/fail status
- Platform and brand information
- Non-sensitive configuration values
- Code file names and coverage metrics
- Execution times and performance metrics

10. Complete Event Types Reference

Event Type	Description
`Unified_QE_TestExecution_v1`	Test execution results and metadata
`Unified_QE_TestCaseMapping_v2`	Test case to code file mappings
`Unified_QE_TotalTestCoverage_v1`	Overall code coverage metrics
`Unified_QE_UncoveredFiles_v1`	Files with no test coverage
`Unified_QE_TestExecutorDetails_v1`	Test executor information
`Unified_QE_CoverageFileLinks_v1`	Links to coverage report files
`Unified_QE_CoverageReportLinks_v1`	Links to coverage dashboards
`Unified_QE_AppConfig_Summary_v2`	App configuration summary
`Unified_QE_AppConfig_Value_v2`	Individual config key-value pairs
`Unified_QE_TestRail_v1`	TestRail test case data

`Unified_QE_TestRun_v1`	TestRail test run data
`Unified_QE_Jira_v1`	Jira bug data

11. Conclusion

This internal documentation provides comprehensive technical details for implementing, maintaining, and troubleshooting the New Relic integration. The integration is a critical component of our test analytics infrastructure, providing real-time insights into test quality and code coverage across all supported platforms.

> ### Document Updates
>
> This document should be updated whenever:
>
> - New event types are added
> - API endpoints change
> - Implementation patterns are updated
> - New features are introduced
> - Security policies change

14. Documentation Generation

This documentation was automatically generated using **Cursor AI** (Claude Sonnet 4.5) on October 22, 2025.

The documentation generation process:

- Scanned codebase for New Relic integration patterns
- Analyzed implementation classes and methods
- Generated comprehensive technical documentation
- Created both public and internal versions

To Regenerate: Run `python3 generate_newrelic_docs.py`

Documentation generated with AI assistance using Cursor IDE

Generated on October 22, 2025 at 01:51 PM