**ORACLE-KYRIBA PAYMENT INTEGRATION**

Technical Architecture & Implementation Analysis

Data Flow, API Specifications, and Deployment Strategy

Prepared for: Project Management Office

Version 1.0 | October 2025

**CONFIDENTIAL**

# TABLE OF CONTENTS

1. Executive Summary 3

2. High-Level Architecture 5

3. Technical Components 7

4. Oracle SaaS Adapters 9

5. API Specifications 11

6. Integration Patterns 14

7. Security Architecture 16

8. Data Flow Scenarios 18

9. Error Handling 20

10. Monitoring & KPIs 22

11. Testing Strategy 24

12. Deployment Plan 26

13. Risk Assessment 28

14. Cost Analysis 30

15. Recommendations 32

# 1. EXECUTIVE SUMMARY

## Project Overview

This document presents the comprehensive technical architecture for integrating Oracle Cloud ERP with Kyriba Treasury Management System (TMS) for end-to-end payment processing. The integration will enable automated payment workflows, real-time status updates, and seamless bank connectivity.

### Project Metrics

|  |  |
| --- | --- |
| **Metric** | **Value** |
| Implementation Duration | **12-16 weeks** |
| Integration Pattern | **Hybrid (Real-time API + Batch)** |
| Expected Transaction Volume | **5,000+ payments/day** |
| SLA Target | **99.9% availability** |
| Processing Time Reduction | **85% improvement** |
| Error Rate Reduction | **95% improvement** |
| ROI Period | **1.4 years** |

### Key Benefits

• Automated end-to-end payment processing

• Real-time payment visibility and tracking

• Bank-agnostic payment processing

• Enhanced compliance and audit trails

• Reduced manual intervention and errors

• Global payment capabilities

# 2. HIGH-LEVEL ARCHITECTURE

## 2.1 System Architecture Overview

The integration architecture consists of three primary layers:

• Oracle Cloud ERP Layer: Source system for payment initiation and approval

• Oracle Integration Cloud (OIC): Middleware for transformation and orchestration

• Kyriba TMS Layer: Payment hub for bank connectivity and execution

## 2.2 Data Flow Architecture

|  |  |  |
| --- | --- | --- |
| **Stage** | **System** | **Process** |
| 1 | Oracle ERP | Payment creation and approval in AP module |
| 2 | Oracle ERP | Payment batch formation based on pay groups |
| 3 | OIC | Extract payment data via REST API |
| 4 | OIC | Transform to Kyriba format (ISO 20022) |
| 5 | Kyriba | Validate and enrich payment data |
| 6 | Kyriba | Route to appropriate bank |
| 7 | Bank | Execute payment via SWIFT/ACH/Wire |
| 8 | Kyriba | Receive bank confirmation |
| 9 | OIC | Update Oracle with payment status |
| 10 | Oracle ERP | Update GL and cash position |

# 3. TECHNICAL COMPONENTS

## 3.1 Oracle Cloud ERP Components

|  |  |
| --- | --- |
| **Component** | **Purpose** |
| Accounts Payable | Payment initiation and vendor management |
| Cash Management | Bank account management and reconciliation |
| General Ledger | Accounting entries and financial reporting |
| Oracle Integration Cloud | Middleware and orchestration |
| Enterprise Scheduler | Batch job scheduling and automation |
| Business Events | Real-time event streaming |

## 3.2 Kyriba Components

• Payment Hub: Central payment processing and routing

• Bank Connectivity Module: Multi-bank integration (SWIFT, API, Host-to-Host)

• Cash Management: Liquidity and cash position tracking

• Compliance Engine: OFAC screening and sanctions checking

• FX Module: Foreign exchange rate management

• Reporting Dashboard: Analytics and payment tracking

# 4. ORACLE SAAS ADAPTERS

## 4.1 Required Adapters

The following Oracle Integration Cloud adapters are required for the Kyriba integration:

|  |  |  |
| --- | --- | --- |
| **Adapter** | **Version** | **Purpose** |
| Oracle ERP Cloud | 23.06.1 | Native ERP integration |
| REST Adapter | 23.06.1 | Kyriba API connectivity |
| FTP/SFTP Adapter | 23.06.1 | File-based transfers |
| File Adapter | 23.06.1 | Local file processing |
| Notification | 23.06.1 | Alert and error handling |

## 4.2 Oracle ERP Cloud Adapter Configuration

• Host: https://company.fa.us2.oraclecloud.com

• Authentication: Username Password Token

• Security Policy: Oracle Web Services Security

• Connection Pool: 10 connections

• Timeout: 300 seconds

Key Operations:

• GET /fscmRestApi/resources/payments

• POST /fscmRestApi/resources/paymentProcessRequests

• GET /fscmRestApi/resources/invoices

• POST /fscmRestApi/resources/cashTransactions

• SUBSCRIBE PaymentEvents

# 5. API SPECIFICATIONS

## 5.1 Kyriba REST API Endpoints

|  |  |  |
| --- | --- | --- |
| **Endpoint** | **Method** | **Purpose** |
| /oauth2/token | POST | Authentication |
| /payments/v3/payment-orders | POST | Create payment |
| /payments/v3/payment-orders/{id} | GET | Get payment status |
| /payments/v3/payment-orders/{id}/approve | POST | Approve payment |
| /payments/v3/payment-orders/{id}/cancel | POST | Cancel payment |
| /accounts/v3/bank-accounts | GET | List bank accounts |
| /fx/v3/rates | GET | Get FX rates |
| /webhooks/v3/subscriptions | POST | Register webhooks |

## 5.2 Authentication Flow

OAuth 2.0 Client Credentials Flow:

1. Request token with client\_id and client\_secret

2. Receive access\_token (valid for 3600 seconds)

3. Include token in Authorization header: Bearer {token}

4. Refresh token 5 minutes before expiry

5. Implement retry logic for token refresh failures

## 5.3 Sample Payment Request

ISO 20022 Payment Format:

{  
 "paymentOrder": {  
 "externalReference": "ORA-PMT-2025-001234",  
 "paymentMethod": "ACH",  
 "valueDate": "2025-10-30",  
 "currency": "USD",  
 "amount": 50000.00,  
 "debitAccount": {  
 "accountId": "KYR\_CTB\_USD\_001"  
 },  
 "beneficiary": {  
 "name": "ABC Corporation",  
 "accountNumber": "123456789",  
 "routingNumber": "021000021"  
 }  
 }  
}

# 6. INTEGRATION PATTERNS

## 6.1 Integration Pattern Selection

|  |  |  |
| --- | --- | --- |
| **Pattern** | **Use Case** | **Volume** |
| Real-time API | Urgent wires, Critical payments | < 100/hour |
| Batch File Transfer | Standard ACH, Payroll | 1000-5000/batch |
| Event-Driven | Status updates, Returns | Continuous |
| Hybrid | Mixed payment types | Variable |

## 6.2 Decision Criteria

Pattern selection is based on the following criteria:

• Payment Amount: > $100,000 uses real-time API

• Payment Method: Wire transfers use real-time, ACH uses batch

• Time Sensitivity: Same-day payments use real-time

• Volume: High volume (>100) uses batch processing

• Business Hours: After-hours payments are queued for batch

# 7. SECURITY ARCHITECTURE

## 7.1 Security Layers

|  |  |
| --- | --- |
| **Layer** | **Controls** |
| Network | VPN tunnel, Firewall rules, IP whitelisting, DDoS protection |
| Application | OAuth 2.0, API keys, Role-based access, Session management |
| Data | TLS 1.3 encryption, PGP file encryption, Tokenization, Data masking |
| Audit | Transaction logging, Change tracking, Compliance reporting |

## 7.2 Compliance Requirements

• OFAC Screening: Real-time sanctions checking

• PCI-DSS: Payment card data protection

• SOX: Financial controls and audit trails

• GDPR: Data privacy and protection

• NACHA: ACH payment compliance

• SWIFT: International payment standards

# 8. DATA FLOW SCENARIOS

## 8.1 Real-Time Wire Payment Flow

1. Payment approved in Oracle ERP AP module

2. Payment event triggers OIC integration flow

3. OIC extracts payment details via REST API

4. Data transformed to ISO 20022 format

5. Payment sent to Kyriba via REST API

6. Kyriba validates and enriches payment data

7. Payment routed to bank via SWIFT MT103

8. Bank sends confirmation (MT900)

9. Kyriba webhook updates OIC with status

10. OIC updates Oracle ERP payment status

11. GL entries created for cash movement

## 8.2 Batch ACH Processing Flow

1. Scheduled job triggers at 2:00 PM daily

2. Oracle queries approved payments by pay group

3. Payment batch created (up to 5000 transactions)

4. Batch file generated in NACHA format

5. File encrypted with PGP

6. File transferred to Kyriba via SFTP

7. Kyriba processes and validates batch

8. Payments sent to bank for processing

9. Bank acknowledgment received

10. Status updates sent back to Oracle

11. Reconciliation process initiated

# 9. ERROR HANDLING

## 9.1 Error Categories and Resolution

|  |  |  |
| --- | --- | --- |
| **Error Type** | **Handling Strategy** | **Recovery** |
| API Timeout | Retry with exponential backoff | 3 retries, then queue |
| Authentication Failure | Refresh token and retry | Alert if persistent |
| Validation Error | Log and notify business user | Manual correction |
| Network Failure | Queue for later processing | Auto-retry every 10 min |
| Duplicate Payment | Skip and log warning | Reconciliation report |
| Bank Rejection | Reverse payment in Oracle | Email notification |

## 9.2 Retry Logic Implementation

• Initial retry delay: 5 seconds

• Exponential backoff factor: 2

• Maximum retries: 3

• Maximum delay: 60 seconds

• Circuit breaker threshold: 5 consecutive failures

• Circuit breaker reset: 5 minutes

# 10. MONITORING & KEY PERFORMANCE INDICATORS

## 10.1 Operational KPIs

|  |  |  |
| --- | --- | --- |
| **KPI** | **Target** | **Alert Threshold** |
| Payment Success Rate | > 99.5% | < 98% |
| Average Processing Time | < 30 seconds | > 60 seconds |
| API Response Time (P95) | < 2 seconds | > 5 seconds |
| System Availability | > 99.9% | < 99.5% |
| Error Rate | < 0.5% | > 2% |
| Batch Processing Time | < 10 minutes | > 20 minutes |

## 10.2 Monitoring Dashboard Components

• Real-time payment processing rate

• Payment status distribution (pie chart)

• API latency metrics (line graph)

• Error rate trending (time series)

• Bank connectivity status (health check)

• Queue depth monitoring

• Daily/weekly/monthly volume statistics

# 11. TESTING STRATEGY

## 11.1 Test Phases

|  |  |  |
| --- | --- | --- |
| **Phase** | **Scope** | **Duration** |
| Unit Testing | Individual component validation | 2 weeks |
| Integration Testing | End-to-end flow testing | 2 weeks |
| UAT | Business user validation | 2 weeks |
| Performance Testing | Load and stress testing | 1 week |
| Security Testing | Penetration and vulnerability testing | 1 week |

## 11.2 Test Scenarios

Critical test scenarios to be covered:

• Single payment processing (ACH, Wire, International)

• Batch processing (1000+ payments)

• Payment cancellation and reversal

• Duplicate payment detection

• Error handling and retry logic

• Bank rejection handling

• Reconciliation accuracy

• Performance under peak load

• Failover and recovery

• Security and access control

# 12. DEPLOYMENT PLAN

## 12.1 Environment Strategy

|  |  |  |
| --- | --- | --- |
| **Environment** | **Purpose** | **Configuration** |
| Development | Initial development and unit testing | Single node, Test banks |
| Test | Integration testing | Single node, Test banks |
| UAT | User acceptance testing | HA setup, Test banks |
| Production | Live processing | Multi-AZ HA, Live banks |
| DR | Disaster recovery | Standby, Ready for failover |

## 12.2 Deployment Timeline

Week 1-2: Environment setup and connectivity

Week 3-4: Security configuration and certificates

Week 5-8: Core integration development

Week 9-10: Advanced features and error handling

Week 11-12: Testing and bug fixes

Week 13-14: UAT execution

Week 15: Production deployment preparation

Week 16: Go-live and hypercare

# 13. RISK ASSESSMENT

## 13.1 Technical Risks

|  |  |  |
| --- | --- | --- |
| **Risk** | **Impact** | **Mitigation** |
| API Rate Limiting | High | Implement throttling and batch processing |
| Token Expiry Issues | Medium | Proactive refresh with 5-minute buffer |
| Network Latency | Medium | Deploy in same region, use dedicated circuits |
| Data Format Mismatch | High | Comprehensive validation and mapping |
| Kyriba Downtime | High | Queue mechanism with automatic retry |
| Duplicate Payments | Critical | Idempotency keys and detection logic |

# 14. COST ANALYSIS

## 14.1 Implementation Costs

|  |  |  |
| --- | --- | --- |
| **Category** | **Item** | **Cost** |
| Licensing | Oracle Integration Cloud | $3,000/month |
| Licensing | Additional Adapters | $500/month |
| Infrastructure | Compute & Storage | $450/month |
| Development | Technical Team (4 FTE) | $307,200 |
| Development | Architects (2 FTE) | $192,000 |
| Testing | Test Team (2 FTE) | $64,000 |
| Management | Project Manager | $83,200 |
|  | Total Implementation | **$646,400** |
|  | Annual Operating Cost | **$54,000** |

## 14.2 Return on Investment

• Current manual processing cost: $500,000/year

• Automated processing cost: $54,000/year

• Annual savings: $446,000

• Payback period: 1.4 years

• 5-year NPV: $1.8 million

# 15. RECOMMENDATIONS

## 15.1 Critical Success Factors

1. Early Kyriba Engagement: Schedule technical workshops and confirm API capabilities

2. Security First Approach: Complete security assessment in week 1

3. Phased Rollout: Start with low-risk payments and single country

4. Testing Investment: Allocate 30% of timeline to comprehensive testing

5. Change Management: Develop training plan and communication strategy

6. Bank Participation: Include banks in UAT for end-to-end validation

## 15.2 Immediate Next Steps

• Approve technical architecture and approach

• Finalize Kyriba contract and API access

• Allocate development resources (6 FTE)

• Schedule project kickoff meeting

• Begin environment setup and connectivity

• Engage security team for assessment

• Identify UAT participants from business

• Establish steering committee

## 15.3 Key Decisions Required

|  |  |
| --- | --- |
| **Timeline** | **Decision Required** |
| Week 1 | Integration pattern selection (Real-time vs Batch) |
| Week 1 | Environment strategy approval |
| Week 2 | Security model and compliance approach |
| Week 2 | Error handling and retry strategy |
| Week 4 | Testing strategy and UAT participants |
| Week 8 | Go-live approach (Big Bang vs Phased) |
| Week 12 | Production cutover date |
| Week 14 | Hypercare support model |

# CONCLUSION

The Oracle-Kyriba payment integration represents a strategic initiative that will transform payment processing capabilities. Key benefits include:

• 85% reduction in payment processing time

• 95% reduction in payment errors

• Real-time payment visibility across all banks

• Bank-agnostic payment processing

• Enhanced compliance and audit capabilities

• Scalability to handle 10,000+ payments daily

The recommended hybrid integration approach provides optimal balance between real-time processing for critical payments and efficient batch processing for high-volume transactions.

**With proper planning and execution, this integration will deliver significant operational efficiencies and cost savings with a payback period of less than 18 months.**

────────────────────────────────────────────────────────────────────────────────────────────────────

Document Version: 1.0

Prepared by: Technical Architecture Team

Date: October 2025

Classification: CONFIDENTIAL - Internal Use Only

Distribution: Project Manager, Steering Committee, Technical Team