JWKS to MarkLogic Integration - Complete Usage Guide

⚠️ DISCLAIMER

IMPORTANT: This software is NOT an official Progress MarkLogic product.

- This integration toolset is provided "AS IS" without any warranties or guarantees
- Usage is solely at your own risk
- No support will be provided by Progress MarkLogic for these scripts
- Users are responsible for testing and validating functionality in their environment
- Always test in a development environment before using in production
- Review and understand the code before executing in your infrastructure

By using these scripts, you acknowledge and accept full responsibility for any consequences resulting from their use.

Overview

This guide covers two powerful scripts designed to manage JWT public keys between JWKS (JSON Web Key Set) endpoints and MarkLogic External Security profiles. These tools automate the synchronization of cryptographic keys used for JWT token validation.

Scripts Included

1. **`scripts/extract-jwks-keys.sh`** - Extracts and uploads new keys from JWKS endpoints to MarkLogic
2. **`scripts/cleanup-obsolete-jwks-keys.sh`** - Analyzes and removes obsolete keys from MarkLogic

Prerequisites

System Requirements

- **Operating System**: macOS, Linux, or Unix-like system
- **Shell**: bash (version 4.0 or later)
- **Network Access**: Connectivity to both JWKS endpoints and MarkLogic server

Required Tools

All tools are commonly available and can be installed via package managers:

```
# macOS (using Homebrew)
brew install curl jq python3

# CentOS/RHEL
sudo yum install curl jq openssl python3
```

Tool Descriptions:

- **curl** - HTTP client for API requests
- **jq** - JSON processor for parsing responses
- **openssl** - Cryptographic toolkit for PEM conversion
- **python3** - For ASN.1 DER encoding (RSA key structure creation)

MarkLogic Configuration

- MarkLogic server running and accessible
- External Security profile configured for OAuth/JWT
- Admin credentials with permission to manage External Security

Script 1: scripts/extract-jwks-keys.sh

Purpose

Extracts RSA public keys from JWKS endpoints, converts them to PEM format, and uploads them to MarkLogic External Security profiles. Includes intelligent duplicate detection to prevent adding

existing keys.

Usage Syntax

```
./scripts/extract-jwks-keys.sh <JWKS_ENDPOINT_URL> [--upload-to-marklogic] [OPTIONS]
```

Parameters

Required

- **`<JWKS_ENDPOINT_URL>`** - The HTTPS/HTTP URL of the JWKS endpoint

Flags

- **`--upload-to-marklogic`** - Upload new keys to MarkLogic (default: analysis only)

MarkLogic Configuration Options

- **`--marklogic-host HOST`** - MarkLogic server hostname (default: your-marklogic-server.com)
- **`--marklogic-port PORT`** - MarkLogic Management API port (default: 8002)
- **`--marklogic-user USER`** - MarkLogic admin username (default: admin)
- **`--marklogic-pass PASS`** - MarkLogic admin password (default: your-admin-password)
- **`--external-security NAME`** - External Security profile name (default: Your-External-Security-Profile)

Examples

Basic Key Extraction (Analysis Only)

```
# Extract and display keys without uploading
./scripts/extract-jwks-keys.sh https://auth.example.com/.well-known/jwks.json
```

Output:

```
DCYjkxqf7xLskUc_9tlaJ8-2QC4Vx-G1nPC1qPQro1Q
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAuekhsx8JZydBcsINa2Gt...
-----END PUBLIC KEY-----

9xJihMI2qDH9Aon014odONQLByhF553_8Wu_zcM2StM
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA8I5S+2YnzUeYsV0K0b8A...
-----END PUBLIC KEY-----
```

Upload New Keys to MarkLogic

```
# Extract and upload new keys (skips duplicates)
# Upload new keys to MarkLogic
./scripts/extract-jwks-keys.sh https://auth.example.com/.well-known/jwks.json --upload-
```

Output with Duplicate Detection:

```
🔍 Checking existing keys in MarkLogic External Security profile 'OAUTH2-Keycloak'...
📊 Found 2 existing key(s) in MarkLogic:
   - DCYjkxqf7xLskUc_9tlaJ8-2QC4Vx-G1nPC1qPQro1Q
   - 9xJihMI2qDH9Aon014odONQLByhF553_8Wu_zcM2StM

⏭️  Key 'DCYjkxqf7xLskUc_9tlaJ8-2QC4Vx-G1nPC1qPQro1Q' already exists in MarkLogic - ski
✅ Key 'NewKeyId123' is new - will be added to MarkLogic

🔄 Preparing to upload 1 new key(s) to MarkLogic...
✅ Successfully uploaded JWT secrets to MarkLogic!
```

Advanced Usage with Custom Configuration

Custom MarkLogic Configuration:

```
# Upload with custom MarkLogic server settings
./scripts/extract-jwks-keys.sh https://auth.example.com/jwks \
  --upload-to-marklogic \
  --marklogic-host ml.company.com \
  --marklogic-port 8002 \
  --external-security OAuth2-Production
```

Using Environment Variables for Security:

```
# Set sensitive credentials in environment variables
export ML_USER="your-admin-user"
export ML_PASS="your-secure-password"

# Use variables to avoid exposing credentials in command history
./scripts/extract-jwks-keys.sh https://auth.example.com/jwks \
  --upload-to-marklogic \
  --marklogic-host ml.company.com \
  --marklogic-user "$ML_USER" \
  --marklogic-pass "$ML_PASS" \
  --external-security OAuth2-Production
```

Multiple Environment Support:

```
# Development environment
./scripts/extract-jwks-keys.sh https://dev-idp.company.com/jwks \
  --upload-to-marklogic \
  --marklogic-host dev-ml.company.com \
  --external-security OAuth2-Dev

# Production environment
./scripts/extract-jwks-keys.sh https://prod-idp.company.com/jwks \
  --upload-to-marklogic \
  --marklogic-host prod-ml.company.com \
  --external-security OAuth2-Prod
```

Common JWKS Endpoints

- **Keycloak**: `https://your-keycloak.example.com/realms/{realm}/protocol/openid-connect/certs`
- **Auth0**: `https://your-domain.auth0.com/.well-known/jwks.json`
- **Azure AD**: `https://login.microsoftonline.com/{tenant}/discovery/v2.0/keys`
- **Google**: `https://www.googleapis.com/oauth2/v3/certs`
- **Okta**: `https://your-domain.okta.com/oauth2/default/v1/keys`

Key Features

- **PEM Conversion**: Automatically converts base64URL keys to proper PEM format

- **Duplicate Detection**: Prevents adding keys that already exist in MarkLogic
- **SSL Support**: Handles self-signed certificates with `-k` flag
- **Error Handling**: Comprehensive validation and error reporting
- **Multiple Key Types**: Supports RSA, EC, and symmetric keys (uploads RSA only)

Script 2: scripts/cleanup-obsolete-jwks-keys.sh

Purpose

Compares keys in MarkLogic External Security profiles with current JWKS endpoints to identify and optionally remove obsolete keys that are no longer active.

Usage Syntax

```
./scripts/cleanup-obsolete-jwks-keys.sh <JWKS_ENDPOINT_URL> [--delete-keys] [OPTIONS]
```

Parameters

Required

- **`<JWKS_ENDPOINT_URL>`** - The HTTPS/HTTP URL of the JWKS endpoint

Modes

- **(default)** - Analysis mode - identifies obsolete keys but doesn't delete them
- **`--delete-keys`** - Delete mode - actually removes obsolete keys from MarkLogic

MarkLogic Configuration Options

- **`--marklogic-host HOST`** - MarkLogic server hostname (default: your-marklogic-server.com)
- **`--marklogic-port PORT`** - MarkLogic Management API port (default: 8002)
- **`--marklogic-user USER`** - MarkLogic admin username (default: admin)
- **`--marklogic-pass PASS`** - MarkLogic admin password (default: your-admin-password)
- **`--external-security NAME`** - External Security profile name (default: Your-External-Security-Profile)

Two-Step Workflow

Step 1: Analysis Mode (Safe)

```
# Identify obsolete keys without making changes
./scripts/cleanup-obsolete-jwks-keys.sh https://auth.example.com/.well-known/jwks.json
```

Sample Output:

```
🔍 JWKS Key Cleanup Analysis
================================
JWKS Endpoint: https://auth.example.com/.well-known/jwks.json
MarkLogic External Security Profile: OAUTH2-Keycloak
Mode: ANALYSIS MODE - Will identify obsolete keys only

✅ Found 2 key(s) in current JWKS:
   - NewKeyId123
   - AnotherKeyId456

✅ Found 3 key(s) in MarkLogic:
   - OldKeyId789
   - NewKeyId123
   - AnotherKeyId456

🔄 SYNCHRONIZED KEYS: 2
🆕 MISSING KEYS: 0
🗑️  OBSOLETE KEYS: 1

🗑️  OBSOLETE KEYS (Present in MarkLogic but not in current JWKS):
   Count: 1
   🔴 OldKeyId789

🧹 Cleanup Options:
   4. To delete obsolete keys, run:
      ./scripts/cleanup-obsolete-jwks-keys.sh https://auth.example.com/.well-known/jwks
```

Step 2: Deletion Mode (Destructive)

```
# Remove obsolete keys with confirmation
./scripts/cleanup-obsolete-jwks-keys.sh https://auth.example.com/.well-known/jwks.json
```

Interactive Deletion Process:

```
⚠️  DELETE MODE ENABLED - Obsolete keys will be removed from MarkLogic!

🗑️  DELETING OBSOLETE KEYS
==========================
About to delete 1 obsolete key(s) from MarkLogic...

⚠️  WARNING: This action cannot be undone!
Keys to be deleted:
   🔴 OldKeyId789

Are you sure you want to delete these keys? (yes/no): yes

🗑️  Deleting key: OldKeyId789
   ✅ Successfully deleted key: OldKeyId789

📊 DELETION SUMMARY
==================
✅ Successfully deleted: 1 key(s)
❌ Failed to delete: 0 key(s)
🎉 All obsolete keys have been successfully removed from MarkLogic!
```

Advanced Usage with Custom Configuration

Custom MarkLogic Configuration:

```
# Delete obsolete keys with custom configuration
./scripts/cleanup-obsolete-jwks-keys.sh https://auth.example.com/jwks \
  --delete-keys \
  --marklogic-host ml.company.com \
  --external-security OAuth2-Production
```

Using Environment Variables for Security:

```
# Set sensitive credentials in environment variables
export ML_USER="your-admin-user"
export ML_PASS="your-secure-password"

# Analyze with secure credentials
./scripts/cleanup-obsolete-jwks-keys.sh https://auth.example.com/jwks \
  --marklogic-host ml.company.com \
  --marklogic-user "$ML_USER" \
  --marklogic-pass "$ML_PASS" \
  --external-security OAuth2-Production
```

Safety Features

- **Confirmation Required**: Must type "yes" to proceed with deletion
- **Analysis First**: Encourages running analysis mode before deletion
- **Detailed Warnings**: Shows exactly which keys will be affected
- **Cancellation Support**: Easy to abort the process
- **Error Reporting**: Clear feedback on successes and failures

Configuration

MarkLogic Configuration

Both scripts support flexible configuration through command-line parameters, allowing you to override default settings without modifying the scripts.

Configuration Options

All scripts accept these MarkLogic configuration parameters:

- **`--marklogic-host HOST`** : MarkLogic server hostname or IP address
- **`--marklogic-port PORT`** : Management API port (typically 8002)
- **`--marklogic-user USER`** : Admin user with External Security management permissions
- **`--marklogic-pass PASS`** : Admin password
- **`--external-security NAME`** : Name of the External Security profile configured for JWT

Configuration Methods

1. Command-Line Parameters (Recommended):

```
./scripts/extract-jwks-keys.sh https://idp.company.com/jwks \
  --upload-to-marklogic \
  --marklogic-host ml.company.com \
  --marklogic-user admin \
  --marklogic-pass secret123 \
  --external-security OAuth2-Production
```

2. Environment Variables (Most Secure):

```
export ML_HOST="ml.company.com"
export ML_USER="admin"
export ML_PASS="secret123"
export ML_PROFILE="OAuth2-Production"

./scripts/extract-jwks-keys.sh https://idp.company.com/jwks \
  --upload-to-marklogic \
  --marklogic-host "$ML_HOST" \
  --marklogic-user "$ML_USER" \
  --marklogic-pass "$ML_PASS" \
  --external-security "$ML_PROFILE"
```

3. Default Values (Fallback):

If no parameters are provided, scripts use these defaults:

- Host: your-marklogic-server.com
- Port: 8002
- User: admin
- Password: your-admin-password
- Profile: Your-External-Security-Profile

SSL Certificate Handling

Both scripts include SSL certificate bypass for development environments with self-signed certificates. The `-k` flag is used with curl commands.

For Production: Remove the `-k` flag and ensure proper SSL certificates are configured.

Workflows and Best Practices

Complete Key Lifecycle Management

1. Initial Setup

```
# First time: Extract and upload all current keys
./scripts/extract-jwks-keys.sh https://your-idp.com/jwks \
  --upload-to-marklogic \
  --marklogic-host your-ml-server.com \
  --marklogic-user admin \
  --marklogic-pass your-admin-password \
  --external-security OAuth2-Production
```

2. Key Rotation Handling

```
# After IdP key rotation: Add new keys (automatic duplicate detection)
./scripts/extract-jwks-keys.sh https://your-idp.com/jwks \
  --upload-to-marklogic \
  --marklogic-host your-ml-server.com \
  --marklogic-user admin \
  --marklogic-pass your-admin-password \
  --external-security OAuth2-Production

# Clean up old keys: Analyze first
./scripts/cleanup-obsolete-jwks-keys.sh https://your-idp.com/jwks \
  --marklogic-host your-ml-server.com \
  --marklogic-user admin \
  --marklogic-pass your-admin-password \
  --external-security OAuth2-Production

# Clean up old keys: Delete after verification
./scripts/cleanup-obsolete-jwks-keys.sh https://your-idp.com/jwks \
  --delete-keys \
  --marklogic-host your-ml-server.com \
  --marklogic-user admin \
  --marklogic-pass your-admin-password \
  --external-security OAuth2-Production
```

3. Scheduled Maintenance

```
#!/bin/bash
# Example maintenance script (maintenance.sh)
JWKS_URL="https://your-idp.com/jwks"

echo "=== Key Synchronization Maintenance ==="
echo "Adding new keys..."
./scripts/extract-jwks-keys.sh "$JWKS_URL" --upload-to-marklogic

echo "Analyzing obsolete keys..."
./scripts/cleanup-obsolete-jwks-keys.sh "$JWKS_URL"

echo "=== Maintenance Complete ==="
```

Key Rotation Best Practices

Grace Period Considerations

When rotating keys, consider the following timeline:

1. **T+0**: New key published in JWKS
2. **T+0**: Add new key to MarkLogic (using extract script)
3. **T+24h**: Applications begin using new key for signing
4. **T+72h**: Analyze for obsolete keys (using cleanup script)
5. **T+168h (1 week)**: Remove obsolete keys after ensuring no old tokens remain

Verification Steps

Before deleting obsolete keys:

1. **Check Token Expiry**: Ensure all tokens signed with old keys have expired
2. **Application Testing**: Verify applications work with current key set
3. **Monitoring**: Check for JWT validation errors in application logs
4. **Backup**: Consider backing up MarkLogic configuration before deletion

Troubleshooting

Common Issues and Solutions

1. Connection Errors

Problem: `Failed to fetch JWKS data from endpoint`

```
# Check network connectivity
curl -I https://your-idp.com/jwks

# Test with verbose output
curl -v https://your-idp.com/jwks
```

Solutions:

- Verify URL is correct and accessible
- Check firewall/proxy settings
- Ensure SSL certificates are valid (or use -k for testing)

2. MarkLogic Authentication Errors

Problem: `XDMP-OAUTH: Access token provided is empty`

```
# Test MarkLogic connectivity
curl -u admin:password http://marklogic-host:8002/manage/v2/external-security
```

Solutions:

- Verify MarkLogic credentials in script configuration
- Ensure admin user has External Security permissions
- Check MarkLogic server is running and accessible

3. JSON Parsing Errors

Problem: `Invalid JSON response from JWKS endpoint`

```
# Validate JWKS response manually
curl https://your-idp.com/jwks | jq .
```

Solutions:

- Verify JWKS endpoint returns valid JSON
- Check for HTML error pages instead of JSON
- Ensure endpoint is the correct JWKS URL

4. Key Format Issues

Problem: Keys not converting to PEM format

```
# Check if required tools are available
which openssl python3
```

Solutions:

- Install missing dependencies (openssl, python3)
- Verify RSA key parameters (n, e) are present in JWKS
- Check for unsupported key types (only RSA keys are uploaded)

5. Permission Issues

Problem: Script execution permission denied

```
# Make scripts executable
chmod +x scripts/extract-jwks-keys.sh
chmod +x scripts/cleanup-obsolete-jwks-keys.sh
```

Debug Mode

For troubleshooting, you can modify the scripts to add debug output:

```
# Add to beginning of script for verbose output
set -x  # Enable debug mode
```

Security Considerations

Key Management Security

1. **Principle of Least Privilege**: Use dedicated service accounts with minimal required permissions
2. **Credential Protection**: Store MarkLogic credentials securely (consider environment variables)
3. **Audit Logging**: Monitor all key addition/deletion operations
4. **Network Security**: Use HTTPS for all communications
5. **Access Control**: Restrict script execution to authorized personnel

Production Deployment Security

```
# Example: Using environment variables for credentials
export MARKLOGIC_USER="service-account"
export MARKLOGIC_PASS="secure-password"
export MARKLOGIC_HOST="your-marklogic-server.com"

# Modify script to use environment variables
MARKLOGIC_USER="${MARKLOGIC_USER:-admin}"
MARKLOGIC_PASS="${MARKLOGIC_PASS:-admin}"
```

Key Validation

Before deleting keys, always verify:

- No active JWT tokens are signed with the key
- Applications have been updated to use new keys
- Sufficient grace period has passed for token expiration

Automation and Integration

Cron Job Example

```
# /etc/crontab entry for daily key synchronization
0 2 * * * /path/to/scripts/extract-jwks-keys.sh https://idp.company.com/jwks --upload-t
```

CI/CD Integration

```yaml
# Example GitHub Actions workflow
name: JWKS Synchronization
on:
  schedule:
    - cron: '0 2 * * *'  # Daily at 2 AM
  workflow_dispatch:

jobs:
  sync-keys:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v2

      - name: Install dependencies
        run: |
          sudo apt-get update
          sudo apt-get install -y curl jq openssl python3

      - name: Sync JWKS keys
        env:
          MARKLOGIC_USER: ${{ secrets.MARKLOGIC_USER }}
          MARKLOGIC_PASS: ${{ secrets.MARKLOGIC_PASS }}
        run: |
          ./scripts/extract-jwks-keys.sh ${{ vars.JWKS_URL }} --upload-to-marklogic
```

Monitoring Integration

```
# Example monitoring script
#!/bin/bash
JWKS_URL="https://idp.company.com/jwks"
WEBHOOK_URL="https://monitoring.company.com/webhook"

# Run analysis
RESULT=$(./scripts/cleanup-obsolete-jwks-keys.sh "$JWKS_URL" 2>&1)

# Check for obsolete keys
if echo "$RESULT" | grep -q "Obsolete keys: [1-9]"; then
    # Send alert
    curl -X POST "$WEBHOOK_URL" \
         -H "Content-Type: application/json" \
         -d '{"message": "Obsolete JWKS keys detected in MarkLogic", "details": "'"$RES
fi
```

API Reference

MarkLogic Management API Endpoints Used

Get External Security Configuration

```
GET /manage/v2/external-security/{name}/properties
Authorization: Basic {credentials}
Accept: application/json
```

Add JWT Secrets

```
POST /manage/v2/external-security/{name}/jwt-secrets
Authorization: Basic {credentials}
Content-Type: application/json

{
  "oauth-server": {
    "oauth-jwt-secret": [
      {
        "oauth-jwt-key-id": "key-id",
        "oauth-jwt-secret-value": "-----BEGIN PUBLIC KEY-----\n..."
      }
    ]
  }
}
```

Delete JWT Secret

```
DELETE /manage/v2/external-security/{name}/jwt-secrets/{key-id}
Authorization: Basic {credentials}
```

JWKS Format Reference

```
{
  "keys": [
    {
      "kty": "RSA",
      "kid": "key-identifier",
      "use": "sig",
      "n": "base64url-encoded-modulus",
      "e": "base64url-encoded-exponent"
    }
  ]
}
```

Version History and Updates

Script Versions

- **v1.0**: Basic JWKS extraction and upload
- **v1.1**: Added duplicate detection
- **v1.2**: Enhanced error handling and SSL support
- **v2.0**: Added cleanup script with deletion capability
- **v2.1**: Improved safety features and user feedback

Compatibility

- **MarkLogic**: Tested with versions 11.x and 12.x
- **Operating Systems**: macOS, Linux
- **Shell**: bash 4.0+

Maintenance

Help

1. **Check Prerequisites**: Ensure all required tools are installed
2. **Review Logs**: Check script output for specific error messages
3. **Test Connectivity**: Verify network access to both JWKS and MarkLogic
4. **Validate Configuration**: Confirm MarkLogic settings and credentials

Regular Maintenance Tasks

- **Weekly**: Review obsolete keys analysis
- **Monthly**: Verify key synchronization accuracy
- **Quarterly**: Update scripts if MarkLogic or IdP configurations change
- **Annually**: Review and update security configurations

Script Updates

To update the scripts:

1. Test new versions in development environment
2. Backup current working versions
3. Update configuration variables as needed
4. Run analysis mode first before enabling destructive operations

Conclusion

These JWKS integration scripts provide a complete solution for managing JWT public keys between Identity Providers and MarkLogic External Security profiles. The combination of automated duplicate detection, comprehensive analysis, and safe deletion processes ensures reliable and secure key lifecycle management.

Key Benefits:

- **Automation**: Reduces manual key management overhead
- **Safety**: Multiple confirmation layers prevent accidental deletions
- **Reliability**: Comprehensive error handling and validation
- **Flexibility**: Works with any JWKS-compliant Identity Provider
- **Security**: Maintains clean, synchronized key sets

For production deployments, always test in development environments first and follow security best practices for credential management and access control.