CLM Enhancement - Quick Reference Guide

🚀 Quick Start

1. Standalone IO API Execution

```java
@Autowired
private IOApiService ioApiService;

// Simple playbook execution
Map<String, String> params = Map.of("key", "value");
IOExecuteRequest request = ioApiService.buildExecuteRequest(
    12345,                          // CSI
    "PROD",                         // Environment
    "server.example.com",           // Target server
    "my_playbook",                  // Playbook name
    "MY_ACTION",                    // Action
    params,                         // Parameters
    UUID.randomUUID().toString()    // Transaction ID
);

IOExecuteResponse response = ioApiService.executePlaybook(request);
// Result tracked automatically in TransactionLogs & AnsibleResultRequest
```

2. Certificate Scanner

```bash
# Manual full scan
curl -X POST http://localhost:8080/api/v1/scanner/scan/trigger \
  -H "X-User-Id: admin"

# Scan specific CSI
curl -X POST http://localhost:8080/api/v1/scanner/scan/csi/12345 \
  -H "X-User-Id: admin"

# Get results
curl http://localhost:8080/api/v1/scanner/scan/latest
curl http://localhost:8080/api/v1/scanner/scan/stats
```

📋 Key Classes

Class	Purpose	Usage
`IOApiService`	Standalone IO execution	Execute any playbook
`CertificateScannerService`	Certificate scanning	CSI-wise batch scanning
`ScannerAdminController`	Scanner APIs	Manual triggers & monitoring
`ResultCallbackControllerEnhanced`	Callback handler	Processes all callback types

🔧 Configuration Quick Reference

```yaml
# Scanner Settings
clm.scanner:
  enabled: true                    # Enable/disable
  scan-cron: "0 0 3 * * ?"         # Daily at 3 AM
  server-batch-size: 20            # Servers per batch
  max-requests-per-minute: 60      # Rate limit
  delay-between-batches-ms: 1000   # Batch delay
  delay-between-csis-ms: 5000      # CSI delay
  scaling-issue-threshold: 10      # Pause threshold
  scan-playbook-name: clm_certificate_scan  # Playbook

# IO API Settings
clm.io-api:
  base-url: https://b2b.cti.otservices.citigroup.net
  basic-auth-credentials: ${IO_API_BASIC_AUTH}
  max-requests-per-minute: 60
```

📊 Database Collections

server_inventory

```javascript
{
  csi: 12345,
  hostname: "server01.example.com",
  connectionStatus: "SUCCESS",       // SUCCESS/FAILED/UNKNOWN
  lastScanDate: ISODate("..."),
  lastScanStatus: "COMPLETED",       // COMPLETED/FAILED/IN_PROGRESS
  certificatesFound: 15,
  active: true
}
```

scan_executions

```javascript
{
  scanDate: ISODate("..."),
  status: "COMPLETED",
  totalCsis: 50,
  processedServers: 1000,
  successfulServers: 980,
  failedServers: 20,
  csiBatches: [...],
  scalingIssues: [...],
  hasScalingIssues: false
}
```

🎯 Common Operations

Populate Server Inventory

```javascript
```

```
db.server_inventory.insertMany([
  {
    csi: 12345,
    hostname: "server01.example.com",
    connectionStatus: "SUCCESS",
    active: true,
    environment: "PROD",
    osType: "Linux",
    productType: "WAS",
    createdDate: new Date()
  }
])
```

Check Active Servers

javascript

```javascript
db.server_inventory.find({
  active: true,
  connectionStatus: "SUCCESS"
}).count()
```

View Latest Scan

javascript

```javascript
db.scan_executions.find().sort({scanDate: -1}).limit(1)
```

Find Scaling Issues

javascript

```javascript
db.scan_executions.find({
  hasScalingIssues: true
}).sort({scanDate: -1})
```

🔍 API Endpoints

Scanner Endpoints

```
POST   /api/v1/scanner/scan/trigger            # Trigger full scan
POST   /api/v1/scanner/scan/csi/{csi}          # Scan CSI
GET    /api/v1/scanner/scan/latest             # Latest scan
GET    /api/v1/scanner/scan/stats              # Statistics
GET    /api/v1/scanner/scan/scaling-issues     # Scans with issues
GET    /api/v1/scanner/servers/csi/{csi}/active  # Active servers
```

Callback Endpoint

```
POST   /api/v1/result                          # Universal callback
```

⚙️ Tuning Guide

Small Environment (<1000 servers)

```yaml
server-batch-size: 50
max-requests-per-minute: 100
delay-between-batches-ms: 500
delay-between-csis-ms: 2000
```

Large Environment (>10000 servers)

```yaml
server-batch-size: 10
max-requests-per-minute: 30
delay-between-batches-ms: 2000
delay-between-csis-ms: 10000
max-servers-per-csi: 200
```

🐛 Troubleshooting

Scanner Taking Too Long

1. Increase `server-batch-size`
2. Increase `max-requests-per-minute`
3. Decrease delays

Rate Limit Errors

1. Decrease `max-requests-per-minute`
2. Increase `delay-between-batches-ms`
3. Decrease `server-batch-size`

Servers Not Scanned

Check:

- `active = true`
- `connectionStatus = SUCCESS`
- Not exceeding `max-servers-per-csi`

📈 Monitoring

Key Metrics

```bash
# Success rate
curl http://localhost:8080/api/v1/scanner/scan/stats | \
  jq '.lastScanServersSuccessful / .lastScanServersProcessed'

# Scaling issues
curl http://localhost:8080/api/v1/scanner/scan/scaling-issues | \
  jq 'length'
```

Log Monitoring

```bash
tail -f logs/clm-service.log | grep "CertificateScannerService"
tail -f logs/clm-service.log | grep "Scaling issue detected"
tail -f logs/clm-service.log | grep "Rate limit reached"
```

📝 TODOs Before Production

1. **Set scan playbook name**

```yaml
  scan-playbook-name: <your_actual_playbook>
```

2. **Populate server inventory**
   - Add all servers with connection status
   - Mark active/inactive
3. **Test with single CSI**

```bash
  curl -X POST .../scanner/scan/csi/12345
```

4. **Monitor first scan**
   - Check scaling issues
   - Adjust rate limits
   - Tune batch sizes
5. **Enable scheduler**

```yaml
  clm.scanner.enabled: true
```

🎓 Learning Path

1. **Read**: `ENHANCEMENT_SUMMARY.md` - Overview
2. **Read**: `SCANNER_DOCUMENTATION.md` - Deep dive
3. **Try**: Manual CSI scan
4. **Review**: Scan results and logs
5. **Tune**: Configuration for your environment
6. **Enable**: Daily scheduler

💡 Pro Tips

1. **Start Conservative**: Begin with low rate limits, increase gradually
2. **Monitor First Scan**: Watch for scaling issues on initial run
3. **Batch by CSI Size**: Large CSIs may need smaller batches
4. **Check Connections**: Ensure server inventory has correct connection status
5. **Use Delays**: Don't rush - stability > speed

🆘 Getting Help

Check Logs:

```bash
tail -f logs/clm-service.log
```

Check Scan Status:

```bash
curl http://localhost:8080/api/v1/scanner/scan/active
```

Review Scaling Issues:

```bash
```

```
curl http://localhost:8080/api/v1/scanner/scan/scaling-issues
```

Database Queries:

```javascript
// Servers not scanned in 7 days
db.server_inventory.find({
  active: true,
  connectionStatus: "SUCCESS",
  lastScanDate: {$lt: new Date(Date.now() - 7*24*60*60*1000)}
})
```

✅ Checklist

Before deployment:

- [ ] IO API credentials configured
- [ ] Server inventory populated
- [ ] Connection status validated
- [ ] Scan playbook name set
- [ ] Rate limits configured
- [ ] Test scan completed
- [ ] Logs reviewed
- [ ] No scaling issues
- [ ] Documentation reviewed
- [ ] Scheduler enabled

📚 Documentation Files

- `ENHANCEMENT_SUMMARY.md` - Complete overview
- `SCANNER_DOCUMENTATION.md` - Scanner deep dive
- `DOCUMENTATION.md` - Original CLM docs
- `README.md` - Project structure
- `QUICK_REFERENCE.md` - This file