BOBO Sync Process Guide

Overview

The BOBO Sync system processes worker duty status CSV files from the BOBO system and synchronizes them with AtHoc emergency notification systems. This guide explains the complete flow through the scripts and provides troubleshooting guidance.

System Architecture

Core Components

1. **`athoc_client.py`** - Generic AtHoc API client library
2. **`bobo_processor.py`** - Main processing engine
3. **SQLite Database** - Local mapping storage (`bobo_mapping.db`)
4. **Configuration** - Environment variables (`.env` file)
5. **Virtual Environment** - Python dependencies (`myenv/` directory)
6. **Network Testing** - Connectivity and SSL testing utilities (`network_testing/` directory)

Project Structure

```
DMS_Python1/
├── bobosync/                # Main application directory
│   ├── athoc_client.py      # AtHoc API client
│   ├── bobo_processor.py    # Main processing engine
│   ├── requirements.txt     # Python dependencies
│   ├── .env                 # Configuration file
│   ├── bobo_mapping.db      # SQLite database
│   ├── myenv/               # Python virtual environment
│   ├── network_testing/     # Network testing utilities
│   ├── packages/            # Offline installation packages
│   ├── run_bobo_windows.bat # Windows batch wrapper
│   └── run_bobo_windows.ps1 # PowerShell wrapper
├── logs/                    # Log files directory
├── processed_files/         # Successfully processed CSV files
└── failed_files/            # Files that exceeded retry attempts
```

Running the Application

Using Virtual Environment (Recommended)

```
# Navigate to the application directory
cd bobosync

# Run using virtual environment Python
myenv\Scripts\python bobo_processor.py
```

```
# Or activate virtual environment first
myenv\Scripts\Activate.ps1
python bobo_processor.py
```

Using Windows Wrapper Scripts

```
# Navigate to the application directory
cd bobosync

# Run using batch wrapper
.\run_bobo_windows.bat

# Or run using PowerShell wrapper
.\run_bobo_windows.ps1
```

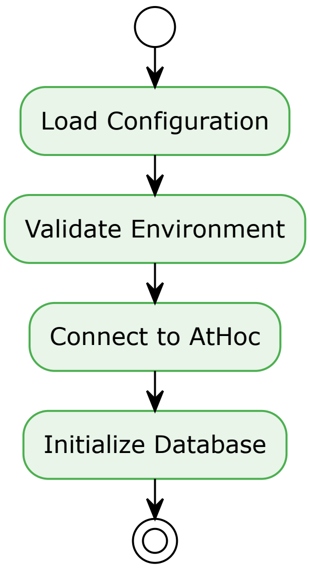
Network Testing Utilities

Before running the main application, you can use the network testing utilities to diagnose connectivity issues:

```
cd bobosync\network_testing

# Test AtHoc authentication
myenv\Scripts\python test_athoc_auth.py

# Test SSL configuration
myenv\Scripts\python test_ssl_verification.py

# Test proxy connection (if applicable)
myenv\Scripts\python test_proxy_connection.py

# Run comprehensive network tests
myenv\Scripts\python test_all_proxy_scenarios.py
```

Detailed Process Flow

1. Initialization Phase



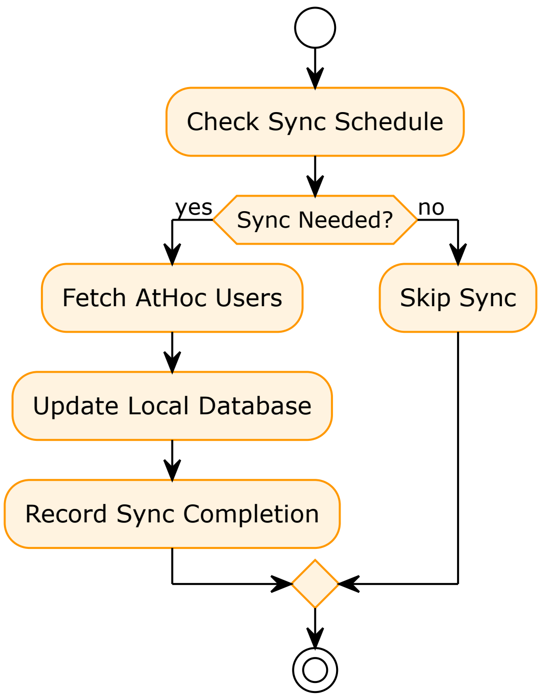
What Happens:

- Loads `.env` file from the same directory as the script
- Validates all required environment variables are present
- Creates a requests session with TLS 1.2 enforcement
- Authenticates with AtHoc OAuth2 server
- Initializes SQLite database for worker mapping

Key Components:

- `BOBOProcessor.__init__()` - Main initialization
- `AtHocClient._get_auth_token()` - Authentication
- `BOBODatabase.init_database()` - Database setup

2. User Mapping Sync Phase



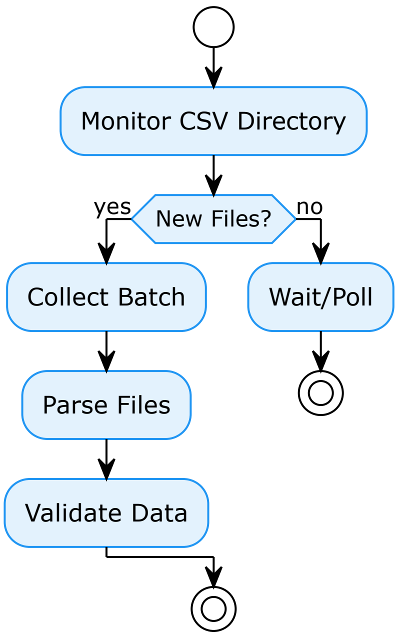
What Happens:

- Checks if daily user mapping sync is due (configurable time)
- Fetches all users with specified attributes from AtHoc
- Updates local SQLite database with employee_id to username mappings
- Records sync completion in tracking table

Key Components:

- `BOBOProcessor.should_run_user_mapping_sync()` - Schedule check
- `AtHocClient.get_all_users_with_attributes()` - AtHoc user fetch
- `BOBODatabase.update_mapping()` - Local database update

3. File Monitoring Phase



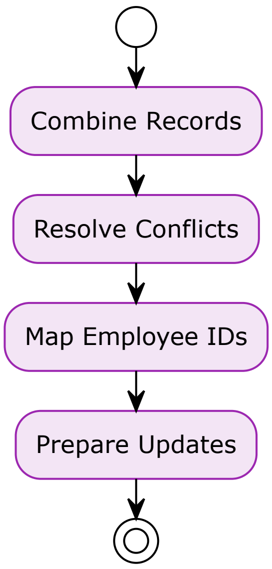
What Happens:

- Continuously monitors the configured CSV directory
- Collects all available CSV files into a single processing batch
- Parses each CSV file into `BOBOEntry` objects
- Validates CSV format and data integrity

Key Components:

- `BOBOProcessor.get_csv_files()` - File discovery
- `BOBOProcessor.parse_csv_file()` - CSV parsing
- `BOBOEntry.from_csv_row()` - Data validation

4. Data Processing Phase



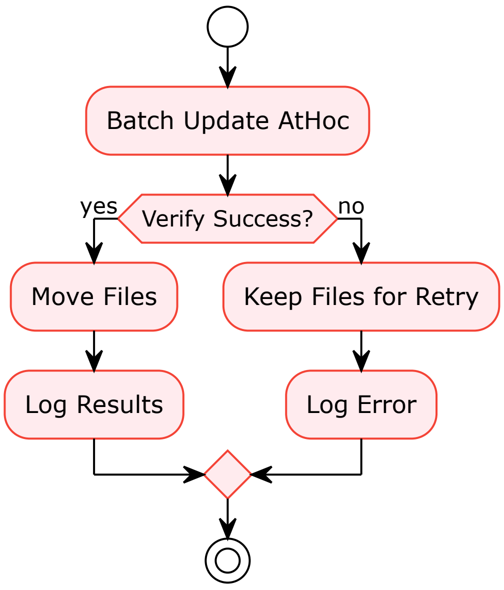
What Happens:

- Combines all CSV records from the batch
- Resolves conflicts (latest timestamp wins for same employee)
- Maps employee IDs to AtHoc usernames using local database
- Prepares duty status updates for AtHoc

Key Components:

- `BOBOProcessor.process_file_batch()` - Main processing logic
- `BOBODatabase.get_username_by_employee_id()` - Username lookup
- Conflict resolution logic (latest timestamp wins)

5. AtHoc Synchronization Phase



What Happens:

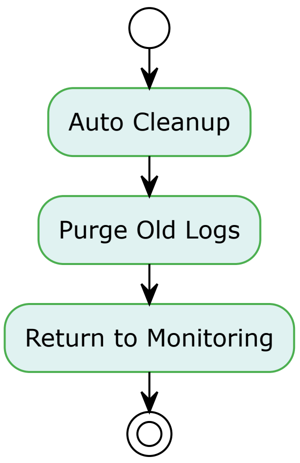
- Sends all duty status updates to AtHoc in a single batch API call
- Verifies successful update from AtHoc response
- Only moves files to processed directory after successful sync

- Logs batch processing results

Key Components:

- `AtHocClient.batch_update_duty_status()` - AtHoc API call
- `BOBOProcessor.move_processed_file()` - File management
- `BOBODatabase.log_processing()` - Result logging

6. Auto-Cleanup Phase



What Happens:

- **Always runs** regardless of whether CSV files are found
- Automatically clears old duty status records from AtHoc (configurable threshold)
- Purges old log files based on retention settings
- Returns to file monitoring phase

Key Components:

- `AtHocClient.clear_old_duty_status()` - AtHoc cleanup
- `AtHocClient.query_users_with_old_duty_status()` - Find old entries
- `AtHocClient.batch_update_duty_status()` - Clear old entries
- `BOBOProcessor._purge_old_logs()` - Log cleanup

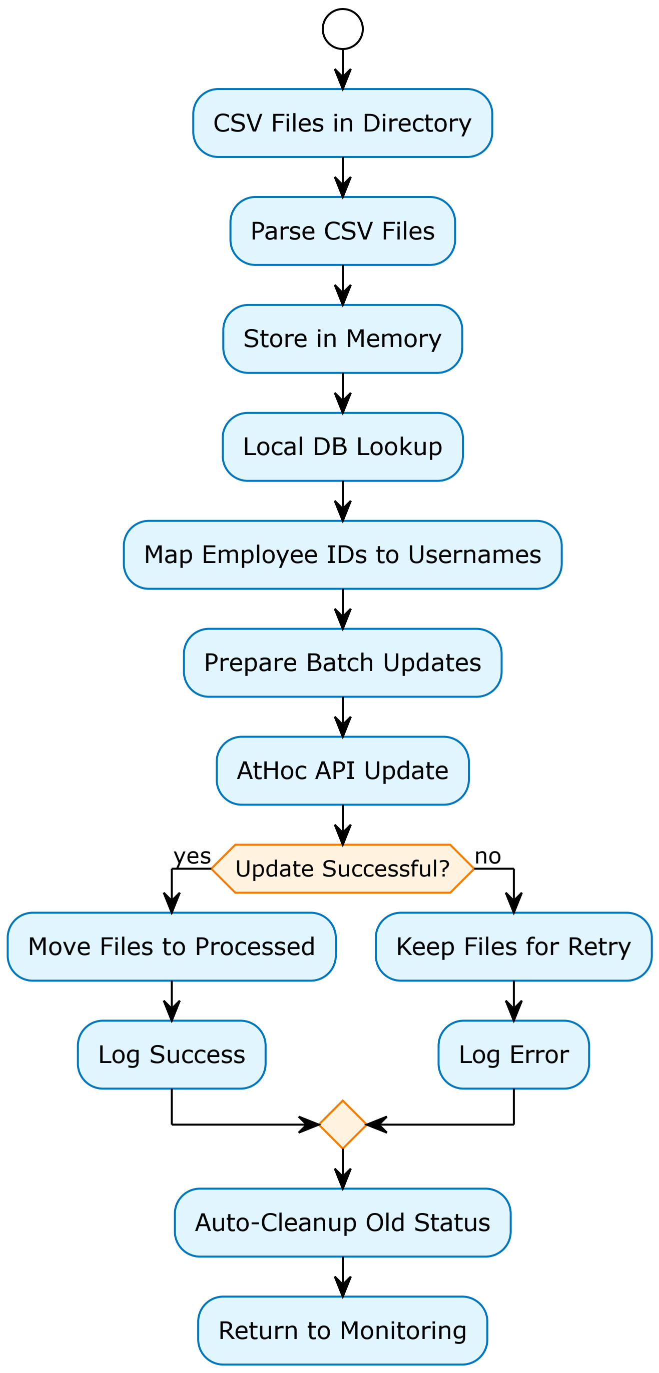
Configuration:

- `AUTO_CLEANUP_HOURS`: Hours after which duty status is considered old (default: 24)
- `DUTY_STATUS_FIELD`: Field name to clear (must match AtHoc configuration)

Benefits:

- Prevents stale duty status data from accumulating
- Maintains data accuracy in AtHoc
- Reduces manual maintenance requirements
- Ensures only current duty status is displayed

Data Flow Diagram



Key Configuration Points

Environment Variables (.env file)

- **AtHoc Connection**: SERVER_URL, CLIENT_ID, CLIENT_SECRET, USERNAME, PASSWORD, ORG_CODE
- **Data Processing**: CSV_DIRECTORY, DUTY_STATUS_FIELD, BATCH_SIZE
- **User Mapping**: USER_ATTRIBUTES, SYNC_HOUR, SYNC_RETRY_DAYS
- **File Management**: PROCESSED_DIRECTORY, MOVE_PROCESSED_FILES
- **Auto-Cleanup**: AUTO_CLEANUP_HOURS, DUTY_STATUS_FIELD
- **Security**: DISABLE_SSL_VERIFY, SCOPE

Database Tables

- **worker_mapping**: employee_id → username mappings
- **processing_log**: File processing history

- **sync_tracking**: User mapping sync status

Troubleshooting Guide

Virtual Environment Issues

Symptoms:

- "ModuleNotFoundError: No module named 'typing_extensions'" or similar
- Import errors for required packages
- Script fails to start

Solution: Always use the virtual environment when running the application:

```
# Navigate to the application directory
cd bobosync

# Use virtual environment Python (recommended)
myenv\Scripts\python bobo_processor.py

# Or activate virtual environment first
myenv\Scripts\Activate.ps1
python bobo_processor.py
```

If virtual environment is missing or corrupted:

```
# Recreate virtual environment
cd bobosync
python -m venv myenv
myenv\Scripts\activate
pip install -r requirements.txt
```

Authentication Issues

Symptoms:

- "Failed to get AtHoc authentication token" error
- HTTP 401 Unauthorized responses
- Connection timeouts during auth

Areas to Check:

1. **Environment Variables**:

   ```
   # Verify all required variables are set
   grep -E "
   (ATHOC_SERVER_URL|CLIENT_ID|CLIENT_SECRET|USERNAME|PASSWORD|ORG_CODE)" .env
   ```

2. **Network Connectivity**:

   ```
   # Test basic connectivity
   curl -I https://your-athoc-server.com
   ```

3. **SSL/TLS Issues**:

   - Check `DISABLE_SSL_VERIFY` setting
   - Verify TLS 1.2 support on target server
   - Look for SSL certificate warnings in logs

4. **Credentials**:

   - Verify username/password are correct
   - Check if account is locked or expired
   - Confirm ORG_CODE matches exactly

Code Locations:

- `87:111:bobosync/athoc_client.py` - Authentication logic
- `95:114:bobosync/athoc_client.py` - Token request

CSV Processing Issues

Symptoms:

- "Error parsing CSV file" messages
- Files not being processed
- Empty batch processing

Areas to Check:

1. **File Format**:

   - Verify CSV has exactly 10 columns
   - Check for proper date/time formats (YYYYMMDD, HHMMSS)
   - Look for missing or extra commas

2. **File Permissions**:

   - Ensure read access to CSV directory
   - Check write access to processed directory

3. **File Naming**:

   - Files must be in CSV_DIRECTORY
   - Must have .csv extension

Code Locations:

- `516:542:bobosync/bobo_processor.py` - CSV parsing

- `47:65:bobosync/bobo_processor.py` - BOBOEntry validation

Database Issues

Symptoms:

- "Database locked" errors
- Missing username mappings
- Sync tracking failures

Areas to Check:

1. **Database File**:
    - Check if `bobo_mapping.db` exists and is writable
    - Verify database isn't corrupted
2. **Mapping Data**:

```sql
-- Check mapping count
SELECT COUNT(*) FROM worker_mapping;

-- Check recent sync
SELECT * FROM sync_tracking WHERE sync_type = 'user_mapping';
```

3. **User Mapping Sync**:
    - Verify `USER_ATTRIBUTES` environment variable
    - Check AtHoc user search API permissions

Code Locations:

- `69:103:bobosync/bobo_processor.py` - Database initialization
- `423:462:bobosync/bobo_processor.py` - User mapping sync

AtHoc API Issues

Symptoms:

- Batch update failures
- HTTP 500/502/503 errors
- Rate limiting messages

Areas to Check:

1. **API Permissions**:
    - Verify user has permission to update duty status
    - Check if duty status field exists in AtHoc
2. **Rate Limiting**:

    - Monitor for HTTP 429 responses
    - Adjust `BATCH_SIZE` if needed

3. **Field Configuration**:

    - Verify `DUTY_STATUS_FIELD` matches AtHoc field name
    - Check field data type and format

Code Locations:

- `491:537:bobosync/athoc_client.py` - Batch update logic
- `567:609:bobosync/bobo_processor.py` - Update processing

Smart Error Handling

New Feature - Intelligent Error Classification:

The system now distinguishes between expected errors and real failures to prevent unnecessary file retries.

Expected Errors (Treated as Success):

- **User Not Found**: When a user doesn't exist in AtHoc (prevents auto-creation)
- **Permission Denied**: When user exists but service account lacks update permissions

Real Errors (Causes Retry):

- **Network Issues**: Connection timeouts, DNS failures
- **API Errors**: Server errors, authentication failures
- **Data Validation**: Invalid field formats, missing required data

Benefits:

- **No False Failures**: Files with only "user not found" errors are processed successfully
- **Reduced Retry Load**: Only real errors cause files to be kept for retry
- **Better Monitoring**: Clear distinction between expected and unexpected issues
- **Data Integrity**: Prevents auto-creation of users while maintaining system reliability

Debug Logging for Error Analysis:

Enable debug logging to see detailed error classification:

```
# Set in .env file
LOG_LEVEL=DEBUG

# Or set environment variable
set LOG_LEVEL=DEBUG
myenv\Scripts\python bobo_processor.py
```

Debug Output Example:

```
Expected: User nonexistent@company.com not found in AtHoc (will be treated as
success)
DEBUG: Treating 'user does not exist' as success for nonexistent@company.com
```

Code Locations:

- `563:584:bobosync/athoc_client.py` - Error classification logic
- `463:474:bobosync/athoc_client.py` - Error logging and categorization

File Management Issues

Symptoms:

- Files not moving to processed directory
- Files being processed multiple times
- Permission denied errors

Areas to Check:

1. **Directory Permissions**:
    - Write access to `PROCESSED_DIRECTORY`
    - Read access to `CSV_DIRECTORY`
2. **Configuration**:
    - Check `MOVE_PROCESSED_FILES` setting
    - Verify directory paths are correct
3. **Disk Space**:
    - Ensure adequate disk space for file operations

Code Locations:

- `474:515:bobosync/bobo_processor.py` - File movement logic

Logging and Monitoring

Log Locations:

- **Main Log**: `../logs/bobo_processor.log`
- **Error Details**: Check log level settings
- **Database Log**: `processing_log` table

Key Log Patterns to Monitor:

- `ERROR` - Critical failures requiring attention
- `WARNING` - Potential issues or missing data
- `Batch processing completed` - Successful operations
- `Authentication successful` - Connectivity confirmation

- `Expected: User not found` - Normal behavior for non-existent users
- `DEBUG: Treating 'user does not exist' as success` - Error classification decisions

Enhanced Debug Mode:

- **Complete JSON Responses**: Full AtHoc API responses with detailed error information
- **Individual User Results**: Per-user sync status, error details, and user IDs
- **Error Classification**: Automatic categorization of expected vs. unexpected errors
- **Decision Logging**: Clear indication of why certain errors are treated as success

Enable Debug Logging:

```
# Set in .env file
LOG_LEVEL=DEBUG

# Or set environment variable
set LOG_LEVEL=DEBUG
myenv\Scripts\python bobo_processor.py
```

Debug Output Example:

```
DEBUG: AtHoc sync_users_by_common_names JSON response: [
  {
    "LOGIN_ID": "user@company.com",
    "On-Duty-DTG": "22/10/2025 07:41:42",
    ":SyncStatus": "OK",
    ":SyncDetails": "Updated",
    "USER_ID": 2060074
  },
  {
    "LOGIN_ID": "nonexistent@company.com",
    "On-Duty-DTG": "22/10/2025 07:41:44",
    ":SyncStatus": "Error",
    ":SyncDetails": "User: nonexistent@company.com does not exists in the
Organization",
    "USER_ID": null
  }
]
Expected: User nonexistent@company.com not found in AtHoc (will be treated as
success)
DEBUG: Treating 'user does not exist' as success for nonexistent@company.com
```

Debug Logging Benefits:

- **Complete Visibility**: See exactly what AtHoc returns for each operation
- **Easy Troubleshooting**: Identify specific issues without guessing
- **Performance Monitoring**: Track API response times and success rates
- **Data Validation**: Verify that expected errors are handled correctly

Performance Issues

Symptoms:

- Slow processing times
- Memory usage growth
- API timeouts

Areas to Check:

1. **Batch Size**:
   - Adjust `BATCH_SIZE` for optimal performance
   - Monitor AtHoc API response times
2. **Database Performance**:
   - Check database file size
   - Consider periodic database maintenance
3. **Memory Usage**:
   - Monitor large CSV file processing
   - Check for memory leaks in long-running processes

Network and Connectivity

Common Issues:

- Firewall blocking AtHoc server
- Proxy configuration problems
- DNS resolution failures
- SSL/TLS certificate issues

Diagnostic Steps:

1. **Use Network Testing Utilities** (Recommended):

   ```
   cd bobosync\network_testing

   # Test AtHoc authentication
   myenv\Scripts\python test_athoc_auth.py

   # Test SSL configuration
   myenv\Scripts\python test_ssl_verification.py

   # Test proxy connection (if applicable)
   myenv\Scripts\python test_proxy_connection.py

   # Run comprehensive network tests
   myenv\Scripts\python test_all_proxy_scenarios.py
   ```

2. **Manual Testing**:

   - Test basic connectivity: `ping athoc-server.com`
   - Test HTTPS: `curl -I https://athoc-server.com`
   - Check proxy settings if applicable
   - Verify firewall rules allow outbound HTTPS

3. **SSL/TLS Issues**:

   - Use `ssl_fix_guide.py` for SSL troubleshooting
   - Check `CUSTOMER_SSL_GUIDE.md` for detailed SSL configuration
   - Use `test_production_ssl.py` for production SSL testing

Auto-Cleanup Issues

Symptoms:

- Old duty status entries not being cleared
- "Auto-cleanup" not running in logs
- Duty status data accumulating over time

Areas to Check:

1. **Configuration**:

   - Verify `AUTO_CLEANUP_HOURS` is set (default: 24)
   - Check `DUTY_STATUS_FIELD` matches AtHoc field name
   - Ensure cleanup runs regardless of CSV file presence

2. **AtHoc Permissions**:

   - Verify user has permission to update duty status fields
   - Check if duty status field exists and is writable
   - Test manual duty status updates

3. **Logging**:

   - Look for "Auto-cleanup" entries in logs
   - Check for cleanup success/failure messages
   - Monitor cleanup operation timing

Code Locations:

- `601:630:bobosync/athoc_client.py` - Auto-cleanup logic
- `555:598:bobosync/athoc_client.py` - Query old duty status
- `506:552:bobosync/athoc_client.py` - Batch update duty status

Manual Testing:

```
# First, navigate to the bobosync directory and activate virtual environment
cd bobosync
myenv\Scripts\activate
```

```python
# Test auto-cleanup functionality
from athoc_client import AtHocClient
client = AtHocClient()

# Query users with old duty status
old_users = client.query_users_with_old_duty_status("DUTY_STATUS", 24)
print(f"Found {len(old_users)} users with old duty status")

# Test cleanup
cleared_count = client.clear_old_duty_status("DUTY_STATUS", 24)
print(f"Cleared duty status for {cleared_count} users")
```

Monitoring Checklist

Daily Checks:

- [ ] User mapping sync completed successfully
- [ ] CSV files being processed without errors
- [ ] No authentication failures
- [ ] Log files within normal size limits
- [ ] Auto-cleanup operations completed successfully
- [ ] No accumulation of old duty status entries

Weekly Checks:

- [ ] Database size reasonable
- [ ] Processed files directory manageable
- [ ] No accumulation of unprocessed files
- [ ] System performance acceptable
- [ ] Auto-cleanup effectiveness (check duty status age distribution)
- [ ] No stale duty status data in AtHoc

Monthly Checks:

- [ ] Review error patterns in logs
- [ ] Validate mapping accuracy
- [ ] Check disk space usage
- [ ] Update credentials if needed

Emergency Procedures

Complete System Failure:

1. **Check Virtual Environment**:

```
cd bobosync
myenv\Scripts\python --version  # Verify Python is working
myenv\Scripts\pip list          # Check installed packages
```

2. **Check AtHoc server status**

3. **Verify network connectivity**:

```
cd bobosync\network_testing
myenv\Scripts\python test_athoc_auth.py
```

4. **Restart the processor service**:

```
cd bobosync
myenv\Scripts\python bobo_processor.py
```

5. **Check recent log entries**

6. **Validate configuration files**

Data Inconsistency:

1. **Stop processing**:

```
# If running in background, stop the process
# Check for running processes
tasklist | findstr python
```

2. **Backup current database**:

```
cd bobosync
copy bobo_mapping.db bobo_mapping_backup_%date%.db
```

3. **Force user mapping sync**:

```
cd bobosync
myenv\Scripts\python -c "from bobo_processor import BOBOProcessor; p =
BOBOProcessor(); p.sync_worker_mappings()"
```

4. **Validate mapping data**

5. **Resume processing with monitoring**:

```
cd bobosync
myenv\Scripts\python bobo_processor.py
```

File Processing Backlog:

1. **Check for processing errors**:

```
cd bobosync
# Check recent logs
type ..\logs\bobo_processor.log | findstr ERROR
```

2. **Verify AtHoc connectivity**:

```
cd bobosync\network_testing
myenv\Scripts\python test_athoc_auth.py
```

3. **Process files manually if needed**:

```
cd bobosync
myenv\Scripts\python bobo_processor.py
```

4. **Monitor batch sizes**

5. **Consider temporary rate limiting**

Auto-Cleanup Failure:

1. **Check AtHoc connectivity and permissions**:

```
cd bobosync\network_testing
myenv\Scripts\python test_athoc_auth.py
```

2. **Verify `AUTO_CLEANUP_HOURS` configuration**:

```
cd bobosync
# Check .env file for AUTO_CLEANUP_HOURS setting
type .env | findstr AUTO_CLEANUP
```

3. **Test manual cleanup using provided code examples**:

    ```
    cd bobosync
    myenv\Scripts\python -c "from athoc_client import AtHocClient; c =
    AtHocClient(); print(f'Cleared: {c.clear_old_duty_status(\"On-Duty-DTG\",
    24)} users')"
    ```

4. **Check for duty status field configuration issues**

5. **Monitor cleanup operation logs for specific errors**:

    ```
    cd bobosync
    type ..\logs\bobo_processor.log | findstr "Auto-cleanup"
    ```

This guide should be referenced whenever issues arise with the BOBO Sync system and updated as new issues or solutions are discovered.