Privacy, Security, Infrastructure Readiness Review for Accreditation API

# Table Of Contents

[Architecture Diagram](#_83njo1smt4my)

[Data Flow - AccreditedEntitiesQueueUpdates](#_w6vzkf3vpxcb)

[Data Flow - AccreditedEntitiesQueueUpdates](#_7u5hqzvwezz0)

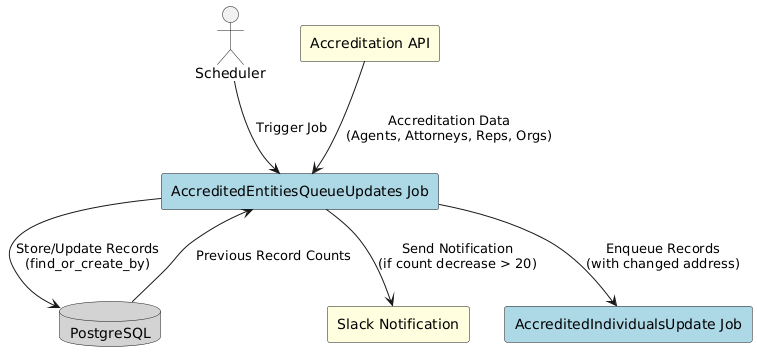
[Architecture Diagram](#_3gvvpg2v77d1)

[Accredited Entities Synchronization Playbook/Incident Response Plan](#_k4bbybtvj1rl)

[Feature Documentation: Representation Management Sidekiq Jobs](#_8ekk02xjkxqw)

# Sequence Diagram

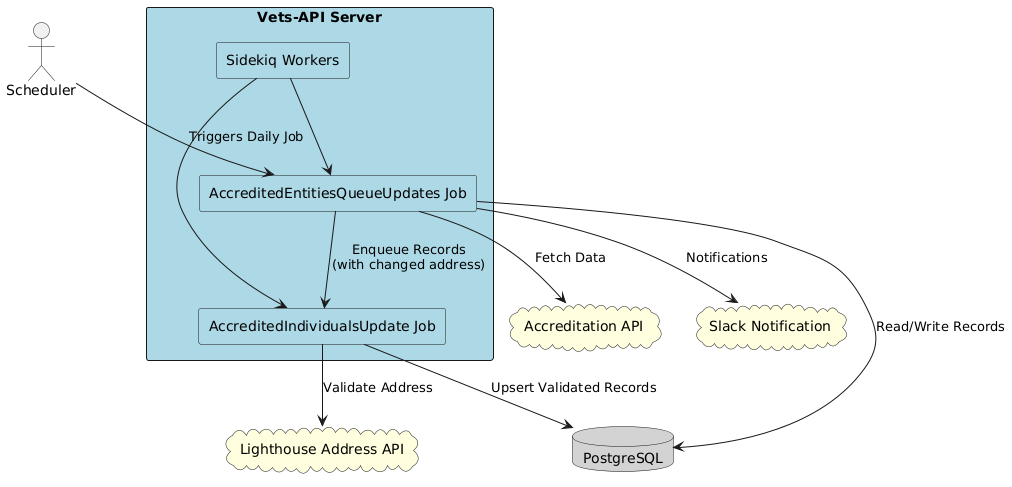
# Data Flow - AccreditedEntitiesQueueUpdates



# 

# Data Flow - AccreditedEntitiesQueueUpdates

# Architecture Diagram



# 

# Accredited Entities Synchronization Playbook/Incident Response Plan

## Product Description

**Product Overview:** The Accredited Entities Synchronization feature is a critical background job system that maintains up-to-date records of accredited representatives (attorneys, agents, VSO representatives) by synchronizing data from the GCLAWS API. This system runs daily and ensures Veterans have access to current information about accredited representatives who can assist them with benefits claims.

**Key Components:**

* Daily synchronization job (AccreditedEntitiesQueueUpdates) running at 4:00 AM ET
* Address validation job (AccreditedIndividualsUpdate) using VAProfile services
* Data integrity checks to prevent unexpected data loss
* Slack notifications for monitoring and alerts

## Contacts

* *All team members can also be reached via the Accredited Representation Management team DSVA Slack channel:* [*#benefits-representation-management*](https://dsva.slack.com/archives/C05L6HSJLHM)
* *Automated notifications are sent to:* [*#benefits-representation-management-notifications*](https://dsva.slack.com/archives/C05L6HSJLHM)

### Team Members

* DSVA Product Lead: Jennifer Bertsch, [jennifer.bertsch@va.gov](mailto:jennifer.bertsch@va.gov)
* Team Product Manager: Lindsay Li-Smith, [lindsay.li-smith@oddball.io](mailto:lindsay.li-smith@oddball.io)
* [Full team roster](https://github.com/department-of-veterans-affairs/va.gov-team/tree/master/products/accredited-representation-management#team-members)

### Outage Contacts:

* Accredited Representation Management team Tech Lead: Holden Hinkle, [holden.hinkle@oddball.io](mailto:holden.hinkle@oddball.io)
* Accredited Representation Management team Backend Engineer: Josh Fike, [josh.fike@oddball.io](mailto:josh.fike@oddball.io)
* Accredited Representation Management team Frontend Engineer: Colin O'Sullivan, [colin.osullivan@adhocteam.us](mailto:colin.osullivan@adhocteam.us)
* Accredited Representation Management team Frontend Engineer: Peri McLaren, [peri.mclaren@adhocteam.us](mailto:peri.mclaren@adhocteam.us)

## Troubleshooting

### Common Issues and Symptoms

#### 1. Count Validation Failures

**Symptoms:**

* Slack alert: "Count decreased by more than X% - skipping update"
* No new data imported despite GCLAWS API being available
* Historical counts showing unexpected drops

**Immediate Actions:**

1. Check GCLAWS API status and recent changes
2. Review AccreditationApiEntityCount records for historical trends
3. Verify if decrease is legitimate (e.g., data cleanup at source)

**Resolution:**

* If decrease is expected: Force update using specific entity types

# Force update for specific entity types

RepresentationManagement::AccreditedEntitiesQueueUpdates.perform\_async(['agents'])

RepresentationManagement::AccreditedEntitiesQueueUpdates.perform\_async(['attorneys'])

# Representatives and VSOs must be processed together

RepresentationManagement::AccreditedEntitiesQueueUpdates.perform\_async(['representatives', 'veteran\_service\_organizations'])

#### 2. Address Validation Failures

**Symptoms:**

* Individual records not updating with geocoding data
* Errors in AccreditedIndividualsUpdate logs
* Records with zero latitude/longitude values

**Immediate Actions:**

1. Check VAProfile service status
2. Review specific address formats causing failures
3. Monitor retry attempts in logs

**Resolution:**

* Check feature flag status: Flipper.enabled?(:remove\_pciu)
* Manually process failed records:

record = AccreditedIndividual.find(id)

RepresentationManagement::AccreditedIndividualsUpdate.new.perform([{

id: record.id,

address: record.raw\_address

}.to\_json])

#### 3. Job Performance Issues

**Symptoms:**

* Jobs taking longer than expected (normal: 2-4 hours)
* Sidekiq queue backing up
* Memory issues on worker nodes

**Immediate Actions:**

1. Check Sidekiq queue depth
2. Monitor worker memory usage
3. Review batch size settings

**Resolution:**

* Adjust SLICE\_SIZE constant if needed (default: 30)
* Scale Sidekiq workers if necessary
* Check for API rate limiting

#### 4. Missing VSO Associations

**Symptoms:**

* Representatives not linked to their VSOs
* Accreditation records missing
* Errors mentioning "VSO not found for ogc\_id"

**Immediate Actions:**

1. Verify VSOs were processed before representatives
2. Check for VSO records in database
3. Review job execution order

**Resolution:**

# Verify VSO exists

vso = AccreditedOrganization.find\_by(ogc\_id: 'vso\_id')

# Check representative associations

rep = AccreditedIndividual.find\_by(ogc\_id: 'rep\_id', individual\_type: 'representative')

rep.accredited\_organizations

# Manually create association if needed

Accreditation.find\_or\_create\_by(

accredited\_individual\_id: rep.id,

accredited\_organization\_id: vso.id

)

### Errors and Metrics

#### Error Logging

* **Rails Logger:** Application logs capture detailed error information
* **Slack Notifications:** Production errors automatically posted to #benefits-representation-management-notifications
* **Sentry Integration:** Runtime errors and exceptions tracked in Sentry

#### Performance Metrics

* **Sidekiq Monitoring:** Monitor job execution times and queue depths
* **Database Metrics:** Track record counts and update rates
* **API Response Times:** Monitor GCLAWS API performance

#### Key Metrics to Monitor

1. **Entity Counts:**
   * Total agents, attorneys, representatives, VSOs
   * Daily count changes
   * Validation threshold breaches
2. **Job Performance:**
   * Total execution time
   * Records processed per minute
   * Address validation success rate
3. **Data Quality:**
   * Records with valid geocoding
   * Failed address validations
   * Missing associations

### Debug Commands

# Check current entity counts from API

RepresentationManagement::AccreditationApiEntityCount.new.api\_counts

# View stored historical counts

AccreditationApiEntityCount.order(created\_at: :desc).limit(7)

# Check job status

Sidekiq::Queue.new.size

Sidekiq::RetrySet.new.size

# Verify data integrity

AccreditedIndividual.where(lat: nil, long: nil).count

AccreditedIndividual.where(individual\_type: 'representative').includes(:accredited\_organizations).where(accredited\_organizations: { id: nil }).count

### Flipper Features and Rollback

* **remove\_pciu**: Toggles between V2 and V3 VAProfile address validation services
  + V2: Uses VAProfile::AddressValidation::Service
  + V3: Uses VAProfile::V3::AddressValidation::Service

### Emergency Procedures

#### Complete Job Failure

1. **Immediate Response:**
   * Check Sidekiq dead set for failed jobs
   * Review error logs for root cause
   * Notify team via Slack
2. **Recovery:**
   * Fix underlying issue
   * Manually trigger job if within business hours
   * Monitor completion

#### Data Corruption

1. **Detection:**
   * Unexpected record deletions
   * Invalid data in updated records
   * Association mismatches
2. **Response:**
   * Stop job execution immediately
   * Restore from database backup if necessary
   * Investigate root cause
   * Implement additional validation

## Scheduled Maintenance

### Daily Operations

* Job runs automatically at 4:00 AM ET
* Monitor Slack channel for completion report
* Review any validation failures

### Weekly Checks

* Review job execution times for trends
* Check address validation success rates
* Verify entity count trends

### Monthly Reviews

* Analyze data quality metrics
* Review and adjust thresholds if needed
* Update documentation as necessary

## Security Considerations

### Data Handling

* No PII/PHI is stored in logs
* Address data is validated but not exposed
* API credentials stored securely in environment variables

### Access Control

* GCLAWS API access restricted by credentials
* VAProfile API access requires authentication
* Database access follows standard Rails security practices

## Configuration Requirements

### Environment Variables

* GCLAWS API credentials
* VAProfile API credentials
* Slack webhook URL for notifications

### Settings

* DECREASE\_THRESHOLD: Maximum allowed percentage decrease (negative value)
* SLICE\_SIZE: Batch size for address validation
* Periodic job schedule in lib/periodic\_jobs.rb

## Dependencies

### External Services

1. **GCLAWS API**
   * Provides accredited entity data
   * Must be accessible from workers
   * Rate limits may apply
2. **VAProfile Address Validation Service**
   * Validates and geocodes addresses
   * Feature flag controls version
   * May have intermittent issues with P.O. Box addresses
3. **Slack Webhooks**
   * Used for notifications
   * Non-critical (job continues if Slack fails)

### Internal Dependencies

* PostgreSQL database
* Sidekiq/Redis
* Rails application framework

## Recovery Procedures

### Restoring from Backup

If data corruption occurs:

1. Identify the last known good state
2. Restore database tables:
   * accredited\_individuals
   * accredited\_organizations
   * accreditations
   * accreditation\_api\_entity\_counts
3. Re-run synchronization job

### Manual Data Recovery

For partial failures:

# Reprocess specific entity types

RepresentationManagement::AccreditedEntitiesQueueUpdates.perform\_async(['agents'])

# Revalidate addresses for specific records

individuals = AccreditedIndividual.where(lat: nil).limit(100)

json\_data = individuals.map do |ind|

{ id: ind.id, address: ind.raw\_address }

end.to\_json

RepresentationManagement::AccreditedIndividualsUpdate.perform\_async(json\_data)

## Post-Incident Review

After any incident:

1. Document timeline and impact
2. Identify root cause
3. Update monitoring/alerts
4. Improve validation logic
5. Update this playbook
6. Share learnings with team

## Additional Resources

* [GCLAWS API Documentation]
* [VAProfile Integration Guide]
* [Sidekiq Best Practices](https://github.com/mperham/sidekiq/wiki/Best-Practices)
* [Team Confluence Space]

# 

# Feature Documentation: Representation Management Sidekiq Jobs

This directory contains Sidekiq background jobs that handle the synchronization and updating of accredited entities (attorneys, agents, VSOs, and representatives) from the GCLAWS API to the local database.

## Overview

The synchronization process consists of two main jobs that work together:

1. **AccreditedEntitiesQueueUpdates** - Fetches and processes entity data from GCLAWS API
2. **AccreditedIndividualsUpdate** - Validates addresses and updates database records

## Jobs

### AccreditedEntitiesQueueUpdates

**Location:** app/sidekiq/representation\_management/accredited\_entities\_queue\_updates.rb

**Schedule:** Daily at 4:00 AM ET (cron: 0 4 \* \* \*)

This is the primary job that initiates the accredited entities update process. It performs the following tasks:

#### Key Features:

* Fetches accredited entities data from GCLAWS API (agents, attorneys, representatives, VSOs)
* Creates or updates AccreditedIndividual records for agents, attorneys, and representatives
* Creates or updates AccreditedOrganization records for VSOs
* Manages Accreditation join records between representatives and their VSOs
* Implements data validation to prevent large decreases in entity counts (configurable threshold)
* Queues address validation jobs for entities with changed addresses
* Removes records no longer present in the GCLAWS API
* Ensures data integrity by processing VSOs before representatives

#### Usage:

# Process all entity types

RepresentationManagement::AccreditedEntitiesQueueUpdates.perform\_async

# Force update specific entity types (bypasses count validation)

RepresentationManagement::AccreditedEntitiesQueueUpdates.perform\_async(['agents'])

RepresentationManagement::AccreditedEntitiesQueueUpdates.perform\_async(['agents', 'attorneys'])

# Representatives and VSOs MUST be processed together

RepresentationManagement::AccreditedEntitiesQueueUpdates.perform\_async(['representatives', 'veteran\_service\_organizations'])



**Important**: Representatives and VSOs must always be processed together to maintain referential integrity. The system will enforce this requirement.

#### Configuration:

* **DECREASE\_THRESHOLD**: Maximum allowed percentage decrease in entity counts before updates are blocked. This is a negative percentage (e.g., -20 for 20% decrease).
* **SLICE\_SIZE**: Number of records processed in each address validation batch (default: 30)

#### Process Flow:

1. Fetches current counts from GCLAWS API
2. Validates counts against stored historical data
3. For each entity type (if validation passes):
   * Fetches all pages of data from API
   * Creates/updates database records
   * Identifies records needing address validation
   * Queues validation jobs in batches
4. Special handling for VSOs and Representatives:
   * VSOs are always processed before representatives
   * Representative-VSO associations are tracked and stored
   * Accreditation join records are created/updated
5. Removes obsolete records (individuals, organizations, and accreditations)

### AccreditedIndividualsUpdate

**Location:** app/sidekiq/representation\_management/accredited\_individuals\_update.rb

This job handles the address validation and final update of AccreditedIndividual records.

#### Key Features:

* Validates addresses using VAProfile Address Validation Service
* Implements retry logic for failed validations
* Handles P.O. Box addresses with special validation logic
* Updates records with validated address data and geocoding information
* Logs errors to Rails logger and Slack (production only)

#### Usage:

This job is automatically queued by AccreditedEntitiesQueueUpdates and should not be called directly.

# Called internally with JSON data

RepresentationManagement::AccreditedIndividualsUpdate.perform\_async(json\_individuals)

#### Address Validation Logic:

1. Attempts validation with full address
2. If validation fails or returns zero coordinates:
   * Retries with modified address (using different address lines)
   * Attempts up to 3 variations
3. Updates record only if valid coordinates are obtained

#### Error Handling:

* Individual record failures don't halt the entire job
* Errors are logged to Rails logger
* Production errors are sent to Slack channel

## Data Flow

GCLAWS API

│

▼

AccreditedEntitiesQueueUpdates

│

├─► Fetches entity data

├─► Creates/updates records

├─► Identifies address changes

│

▼

AccreditedIndividualsUpdate (batched)

│

├─► Validates addresses via VAProfile

├─► Updates geocoding data

└─► Final record update

## Database Models

### AccreditedIndividual

Stores information about accredited agents, attorneys, and representatives:

* individual\_type: 'attorney', 'claims\_agent', or 'representative'
* ogc\_id: Unique identifier from GCLAWS
* registration\_number: Professional registration number
* poa\_code: Power of Attorney code
* Personal information (name, email, phone)
* Address fields (validated and geocoded)
* raw\_address: Original address JSON from GCLAWS

### AccreditedOrganization

Stores information about Veteran Service Organizations (VSOs):

* ogc\_id: Unique identifier from GCLAWS
* poa\_code: Power of Attorney code
* name: Organization name

### Accreditation

Join table linking representatives to their VSOs:

* accredited\_individual\_id: Foreign key to AccreditedIndividual
* accredited\_organization\_id: Foreign key to AccreditedOrganization

## Monitoring

### Logs

* Rails logger for general information and errors
* Slack notifications for production errors (channel: #benefits-representation-management-notifications)

### Count Validation

The system tracks entity counts to detect potential data quality issues:

* Historical counts stored in AccreditationApiEntityCount
* Updates blocked if counts decrease by more than configured threshold
* Force update option available for manual intervention

## Scheduling

The AccreditedEntitiesQueueUpdates job is scheduled to run daily at 4:00 AM ET via the periodic jobs configuration:

# lib/periodic\_jobs.rb

mgr.register('0 4 \* \* \*', 'RepresentationManagement::AccreditedEntitiesQueueUpdates')



This job will automatically queue the necessary AccreditedIndividualsUpdate jobs for address validation.

## Configuration Requirements

### Environment Variables/Settings:

* VAProfile API credentials for address validation
* Slack webhook URL for notifications (production)
* GCLAWS API access credentials

### Feature Flags:

* remove\_pciu: Toggles between V2 and V3 VAProfile address validation services

## Troubleshooting

### Common Issues:

1. **Count validation failures**:
   * Check GCLAWS API for data issues
   * Use force update if decrease is expected
   * Remember: Representatives and VSOs must be processed together
2. **Address validation failures**:
   * Check VAProfile service status
   * Review address format in GCLAWS data
3. **Slow processing**:
   * Adjust SLICE\_SIZE for batch processing
   * Monitor Sidekiq queue depth
4. **Missing VSO associations**:
   * Ensure VSOs are processed before representatives
   * Check that both entity types are included in the job

### Debug Commands:

# Check current entity counts

RepresentationManagement::AccreditationApiEntityCount.new.api\_counts

# Process specific failed record

record = AccreditedIndividual.find(id)

RepresentationManagement::AccreditedIndividualsUpdate.new.perform([{id: record.id, address: {...}}.to\_json])

# Check representative-VSO associations

rep = AccreditedIndividual.find\_by(ogc\_id: 'rep\_id', individual\_type: 'representative')

rep.accredited\_organizations

# Verify VSO exists

vso = AccreditedOrganization.find\_by(ogc\_id: 'vso\_id')

vso.accredited\_individuals

