ODS Actions

Official Document System

Comprehensive Technical Documentation

Frontend, Backend & Database Information

Generated: 2025-11-12 12:35:35

Table of Contents

1. Introduction

ODS Actions is a modern web application designed for managing and processing official documents. The system provides a comprehensive interface for displaying metadata, sending metadata to the ODS system, and uploading files associated with document symbols. This documentation provides detailed information about the frontend architecture (Vue.js), backend architecture (Flask), and database structure (MongoDB) to help users and developers understand and effectively use the application.

2. Frontend Architecture

2.1 Technology Stack

The frontend is built using modern web technologies: • Vue.js 2: Progressive JavaScript framework for building user interfaces • Bootstrap 5: CSS framework for responsive design • Font Awesome: Icon library for visual elements • Modern CSS: Custom design system with CSS variables • JavaScript ES6+: Modern JavaScript features including async/await

2.2 Application Structure

The frontend application is organized as follows: • Main Component: Single Vue component (ods) managing the entire application • Templates: Jinja2 templates for server-side rendering • Static Assets: CSS and JavaScript files in static directory • State Management: Vue.js reactive data management

2.3 Main Features

2.3.1 Display Metadata Tab

Allows users to search and display metadata for document symbols: • Input: Textarea for entering document symbols (one per line) • Processing: Sends POST request to /loading_symbol endpoint • Output: Displays results in a responsive table with columns: - Document Symbol, Agenda, Session, Distribution, Area, Subjects, Job Number, Title, Publication Date, Release Date • Export: CSV export functionality • Validation: Button disabled when input is empty

2.3.2 Send Metadata Tab

Enables users to send document metadata to the ODS system: • Input: Textarea for entering document symbols • Processing: Sends POST request to /create_metadata_ods endpoint • Output: Displays submission results (created/updated/error messages) • Export: CSV export of submission results • Validation: Real-time input validation

2.3.3 Send Files Tab

Handles file uploads associated with document symbols: • Input: Textarea for entering document symbols • Processing: Sends POST request to /exporttoodswithfile endpoint • Output: Displays upload results for each language (AR, ZH, EN, FR, RU, ES, DE) • Features: Progress tracking, result management • Export: CSV export of upload results

2.3.4 Parameters Tab

System administration features: • Site Management: Add sites with code (3-letter), label, and prefix (2-letter) • User Management: Create users with site assignment and tab permissions • System Logs: View and export activity logs with filtering options • Permissions: Control access to Display,

Create/Update, Send Files, Job Numbers, and Parameters tabs

2.3.5 Change Password Tab

User password management: • Email: Pre-filled with current user email • New Password: Input field with validation (minimum 6 characters) • Confirm Password: Confirmation field • Validation: Ensures passwords match before submission

2.4 User Interface Components

• Notification System: Modern notification manager with success, error, warning, and info types • Theme Toggle: Switch between dark and light themes (persisted in localStorage) • Responsive Tables: Scrollable tables with modern styling • Loading Spinners: Visual feedback during API calls • Form Controls: Modern styled inputs, textareas, and buttons • Tab Navigation: Clean tabbed interface for different functions

2.5 Key JavaScript Functions

Main Vue component methods: • displayMetaData(): Fetches and displays metadata for document symbols • displayResultCreateUpdate(): Sends metadata to ODS system • displayResultSendFile(): Handles file uploads to ODS • addUser(): Creates new user accounts • addSite(): Creates new sites • loadLogs(): Loads system activity logs • loadSites(): Loads available sites • exportTableToCSV(): Exports table data to CSV format • changePasswordFromTab(): Handles password changes • validateDocumentSymbols(): Validates input before processing • formatField(): Formats data for display in tables • formatJobNumbersWithFlags(): Formats job numbers with language indicators

3. Backend Architecture

3.1 Technology Stack

The backend is built using: • Flask 3.0.0: Python web framework • MongoDB (PyMongo): Database driver for MongoDB • DLX Library: Library for interacting with Central Database (undlFiles) • Python Decouple: Environment variable management • Werkzeug: Password hashing and security utilities • Requests: HTTP library for API calls to ODS system

3.2 Application Structure

Backend files organization: • ods/__init__.py: Flask application initialization and route definitions • ods/ods_rutines.py: Core business logic and ODS API integration • ods/config_dlx.py: Configuration for Central Database connections • requirements.txt: Python dependencies

3.3 Core Business Logic

3.3.1 ODS API Integration

Functions for interacting with ODS API: • get_token(): Retrieves authentication token from ODS API • ods_get_loading_symbol(): Fetches document metadata from ODS • ods_get_loading_search(): Searches ODS using job numbers • ods_create_update_metadata(): Creates or updates metadata in ODS • ods_file_upload_simple_file(): Uploads files to ODS • update_one_metadata(): Updates a single metadata field

3.3.2 Central Database Integration

Functions for Central Database (CDB/ME): • get_data_from_cb(): Retrieves document data from Central Database • get_subject(): Gets subject information from authority records • get_tcodes(): Gets T-codes from authority records • File.latest_by_identifier_language(): Downloads files from CDB

3.3.3 Job Number Management

Job number handling system: • get_new_job_number(): Generates new job numbers with prefix • check_if_job_number_exists(): Validates job number availability • check_if_docsymbol_exists(): Checks if document symbol exists in ODS • release_job_number(): Releases unused job numbers • Language mapping: Job numbers assigned by language index (AR=0, ZH=1, EN=2, FR=3, RU=4, ES=5, DE=6)

3.3.4 File Processing

File download and upload workflow: • download_file_and_send_to_ods(): Main function for file operations • Process flow: 1. Validate document symbol exists in ODS 2. Query Central Database for files 3. Download files for each language (AR, ZH, EN, FR, RU, ES, DE) 4. Upload files to ODS with

corresponding job numbers 5. Update release dates if needed 6. Release unused job numbers 7. Clean temporary files

3.4 Authentication & Session Management

User authentication system: • Login: POST /login endpoint validates credentials • Password Hashing: Uses Werkzeug's generate_password_hash and check_password_hash • Session Management: Flask sessions store: - username: User email - prefix_site: Site prefix for job numbers - show_display: Permission for Display tab - show_create_update: Permission for Send Metadata tab - show_send_file: Permission for Send Files tab - show_jobnumbers_management: Permission for Job Numbers tab - show_parameters: Permission for Parameters tab • Logout: Clears session data • Default User: Special user with full permissions

3.5 Logging & Analytics

System logging functions: • add_log(): Records user actions and system events - Stores: user, action, date - Collection: ods_actions_logs_collection • add_analytics(): Records detailed analytics data - Stores: user, action, date, data (operation results) - Collection: ods_actions_analytics_collection

4. Database Schema

4.1 MongoDB Database

Database Name: odsActions Connection: Configured via CONN environment variable Client: PyMongo (MongoClient)

4.2 Collections

4.2.1 Users Collection (ods_actions_users_collection)

```
Schema:
{
  "site": String,                    // Site code (3-letter)
  "email": String,                   // User email (unique identifier)
  "password": String,                // Hashed password (Werkzeug)
  "show_display": Boolean,           // Permission for Display Metadata tab
  "show_create_update": Boolean,     // Permission for Send Metadata tab
  "show_send_file": Boolean,         // Permission for Send Files tab
  "show_jobnumbers_management": Boolean,  // Permission for Job Numbers tab
  "show_parameters": Boolean,       // Permission for Parameters tab
  "creation_date": DateTime          // UTC timestamp
}
```

4.2.2 Sites Collection (ods_actions_sites_collection)

```
Schema:
{
  "code_site": String,               // 3-letter site code (unique)
  "Label_site": String,              // Site label/name
  "prefix_site": String,             // 2-letter prefix for job numbers
  "creation_date": DateTime          // UTC timestamp
}
```

4.2.3 Logs Collection (ods_actions_logs_collection)

```
Schema:
{
  "user": String,                    // User email who performed action
  "action": String,                  // Description of the action
  "date": DateTime                   // UTC timestamp
}

Index: Sorted by date (descending) for display
```

4.2.4 Analytics Collection (ods_actions_analytics_collection)

```
Schema:
{
  "user": String,                    // User email
  "action": String,                  // Action type (e.g., "loading_symbol_endpoint")
  "date": DateTime,                  // UTC timestamp
  "data": Array                      // Detailed operation results/data
}
```

4.2.5 Job Numbers Collection (ods_actions_jobnumbers_collection)

```
Schema:
```

```
{
  "created_date": DateTime,          // Creation timestamp
  "jobnumber_value": String,         // Job number (e.g., "NX900000")
  "docsymbol": String,               // Associated document symbol
  "language": String                 // Language code (AR, ZH, EN, FR, RU, ES, DE)
}

Purpose: Tracks job numbers generated by the system
```

4.2.6 ODS Job Number Collection (ods_jobnumber_collection)

```
Schema:
{
  "jobnumber_value": String          // Job number value
}

Purpose: Tracks job numbers released/not used
Operations: Used for releasing unused job numbers
```

4.3 Central Database (undlFiles)

External MongoDB database for document metadata: • Database: undlFiles (via DLX library) • Connection: Configured in config_dlx.py based on environment • Collections: - bibs: Bibliographic records (MARC format) - auths: Authority records - files: File metadata and URIs Key MARC Fields Used: • 191$a: Document symbol • 191$c: Sessions • 245$a, 245$b, 245$c: Title fields • 091$a: Distribution • 269$a: Publication date • 650$a: Subjects • 991$b: Agendas • 035$a: T-codes (in authority records) • 150$a: Subject headings (in authority records)

5. API Endpoints

5.1 Authentication Endpoints

Endpoint	Method	Description
/	GET/POST	Login page and authentication
/logout	GET	User logout and session cleanup
/change_password	POST	Change user password

5.2 Document Operations Endpoints

Endpoint	Method	Description	Input
/loading_symbol	POST	Load and display metadata	docsymbols (textarea)
/create_metadata_ods	POST	Create/update metadata in ODS	docsymbols1 (textarea)
/exporttoodswithfile	POST	Upload files to ODS	docsymbols2 (textarea)

5.3 Administration Endpoints

Endpoint	Method	Description	Input
/add_user	POST	Create new user	site, email, password, permissions
/add_site	POST	Create new site	code_site, label_site, prefix_site
/get_sites	GET	List all sites	None
/list_sites	GET	List sites (alternative)	None
/display_logs	GET	Retrieve system logs	None

5.4 External ODS API Endpoints

The application integrates with external ODS API: • /api/auth/token: Authentication endpoint (GET) - Returns: Access token for ODS API - Authentication: Basic Auth with base64 encoded credentials • /api/loading/symbol: Document symbol operations (GET/POST) - GET: Retrieve metadata for a symbol - POST: Create or update metadata - Parameters: s (symbol), em (exact match) • /api/loading/search: Search by job number (GET) - Parameter: k (job number/keyword) • /api/loading/file: File upload endpoint (POST) - Multipart form data with JSON metadata and PDF file - Updates release dates automatically if needed

6. User Guide

6.1 Getting Started

1. Access the application URL (typically http://localhost:5000) 2. Login with your credentials (email and password) 3. Upon successful login, you'll see the main interface with tabs based on your permissions 4. Use the theme toggle button to switch between dark and light modes

6.2 Display Metadata

Steps to display metadata: 1. Navigate to the "Display Metadata" tab 2. Paste document symbols in the textarea (one per line) 3. Click "Apply" button (enabled when input is provided) 4. Wait for results to load (progress spinner will show) 5. Review results in the table 6. Export to CSV if needed using "Export to CSV" button 7. Clear results using "Clear" or "Clear List" buttons Tips: • You can process multiple symbols at once • Symbols are automatically cleaned (trimmed and uppercased) • Results show "Not found" for missing data

6.3 Send Metadata

Steps to send metadata: 1. Navigate to the "Send Metadata" tab 2. Paste document symbols in the textarea (one per line) 3. Click "Send" button 4. Wait for processing to complete 5. Review results: - "Metadata created!!!" - New metadata created - "Metadata updated!!!" - Existing metadata updated - "Metadata not found in the Central DB/ME" - Symbol not in CDB - Other error messages indicate specific issues 6. Export results to CSV if needed Important: • Metadata is retrieved from Central Database • Job numbers are automatically generated if needed • Only English title is sent (other languages have empty titles)

6.4 Send Files

Steps to send files: 1. Navigate to the "Send Files" tab 2. Paste document symbols in the textarea (one per line) 3. Click "Send Files" button 4. Wait for processing (this may take time for multiple symbols) 5. Review results for each language: - "downloaded and sent successfully!!!" - File uploaded - "file not found in ME/CDB!" - File missing in Central DB - "docsymbol does not exist!!!" - Symbol not in ODS 6. Export results to CSV if needed Process Details: • Files are downloaded from Central Database for all 7 languages • Each file is uploaded to ODS with corresponding job number • Release dates are updated automatically if needed • Unused job numbers are released • Temporary files are cleaned after processing

6.5 System Parameters (Admin Only)

Site Management: 1. Expand "Sites Management" accordion 2. Enter Site Code (3 letters, e.g., "NYC") 3. Enter Site Label (e.g., "New York") 4. Enter Prefix (2 letters, e.g., "NX") 5. Click "Create Site" User

Management: 1. Expand "Users Management" accordion 2. Select a site from dropdown 3. Enter user email 4. Enter password (minimum 6 characters) 5. Check permissions for tabs you want to enable 6. Click "Create User" System Logs: 1. Expand "System Logs" accordion 2. Use filters to search logs by user, action, or date 3. Click "Clear Filters" to reset 4. Export logs using "Export All Logs to CSV"

6.6 Change Password

Steps to change password: 1. Navigate to the "Change Password" tab 2. Your email is pre-filled 3. Enter new password (minimum 6 characters) 4. Confirm new password 5. Click "Change Password" 6. Success notification will appear Validation: • Passwords must match • Minimum 6 characters required • Password is hashed before storage

7. Technical Details

7.1 Environment Variables

Required environment variables (configured via .env or config): • BASE_URL: ODS API base URL • USERNAME: ODS API username • PASSWORD: ODS API password • CLIENT_ID: ODS API client ID • CLIENT_SECRET: ODS API client secret • CONN: MongoDB connection string • DEFAULT_USERNAME: Default admin username • DEFAULT_PASSWORD: Default admin password • DLX_REST_*: Central Database environment variables

7.2 Language Support

The system supports 7 official languages: • AR (Arabic) - Index 0 • ZH (Chinese) - Index 1 • EN (English) - Index 2 • FR (French) - Index 3 • RU (Russian) - Index 4 • ES (Spanish) - Index 5 • DE (German) - Index 6 Job numbers are assigned by language index in arrays.

7.3 File Processing Details

File naming convention: • Format: {docsymbol}_{language}.pdf • Example: A_RES_68_123-EN.pdf Temporary storage: • Windows: ods\temp • Unix/Linux: ./ods/tmp • Files are automatically cleaned after processing File download: • Files are retrieved from Central Database using DLX File API • Files are streamed and saved to temporary directory • Original URIs are used for download File upload: • Multipart form data with JSON metadata • File name matches job number (e.g., "NX900000.pdf") • Release dates updated via PATCH request if needed

7.4 Error Handling

Frontend: • Try-catch blocks around all API calls • User-friendly error notifications • Validation before API calls • Graceful degradation for missing data Backend: • Exception handling in all routes • Error logging to database • Meaningful error messages returned to frontend • Fallback values for missing data Common Error Scenarios: • Symbol not found: Returns "Not found" in results • API errors: Logged and user notified • File not found: Reported in results • Authentication failures: Redirected to login

7.5 Security Features

• Password Hashing: Werkzeug password hashing (PBKDF2) • Session Management: Flask secure sessions • Input Validation: Server-side and client-side validation • Permission System: Tab-level access control • HTTPS Support: Configured for production • Environment Variables: Sensitive data in environment variables • SQL Injection Prevention: Using parameterized queries (MongoDB)

7.6 Performance Considerations

• Sequential Processing: Files processed one language at a time • Batch Operations: Multiple symbols processed in single request • Progress Indicators: Visual feedback during long operations • Temporary File Cleanup: Automatic cleanup after processing • Database Indexing: Logs sorted by date for efficient queries • Connection Pooling: MongoDB connection reuse Optimization Opportunities: • Parallel file processing (future enhancement) • Caching frequently accessed data • Background job processing for large batches

Conclusion

This documentation provides comprehensive information about the ODS Actions application, covering frontend architecture (Vue.js), backend architecture (Flask), and database structure (MongoDB). The application is designed to streamline official document management with modern technology, providing a user-friendly interface for displaying metadata, sending metadata, and uploading files to the ODS system. For additional support or questions, please refer to the README.md file or contact the development team.