# AssistedDiscovery - Comprehensive User Guide

## Table of Contents

1. Introduction
2. Getting Started
3. Configuration Setup
4. Discovery Module
5. Pattern Identification
6. Workspace Management
7. Pattern Library
8. API Management
9. Advanced Features
10. Troubleshooting
11. Best Practices
12. FAQ

## Introduction

AssistedDiscovery is an intelligent XML pattern analysis application that leverages AI-powered pattern extraction and identification capabilities. The system uses Large Language Models(LLM) to automatically discover, extract, verify, and manage XML patterns from various airline API responses, making it easier to understand and work with complex XML structures.

### Key Features

* AI-Powered Pattern Discovery: Automatically extract patterns from XML files using LLMs.
* Pattern Verification: Validate extracted patterns with AI assistance
* Dual Storage System: Save patterns to personal workspaces or shared team libraries
* Pattern Identification: Identify patterns in unknown XML files with airline/version filtering
* Multi-API Support: Work with multiple airline APIs
* Pattern Library Management: Organize and filter patterns efficiently
* Workspace-Based Organization: Separate patterns by use cases and projects
* Interactive Chatbot: Ask Genie questions about analysis results and get intelligent responses
* Modern UI: Clean, professional interface with enhanced user experience

### Target Users

* Airline API Developers: Working with complex XML responses
* Data Analysts: Analyzing XML structure patterns
* QA Engineers: Validating XML data consistency
* Integration Teams: Understanding API response formats

## Getting Started

### System Requirements

* Python: 3.8 or higher
* Streamlit: 1.28.0 or higher
* Azure OpenAI: Active subscription with GPT-4o deployment
* Storage: Minimum 100MB for database files
* Memory: 4GB RAM recommended

### Installation

1. Clone the Repository:

git clone <repository-url>  
 cd AssistedDiscovery

1. Install Dependencies:

pip install -r requirements.txt

1. Launch the Application:

streamlit run app/Home.py

1. Access the Interface:

* Open your browser to http://localhost:8501
* The application will launch with the Home page
* Navigate using the sidebar to Discovery or Identify pages

### Initial Setup Checklist

* [ ] Azure OpenAI credentials configured
* [ ] GPT-4o model deployment verified
* [ ] Test workspace created
* [ ] Sample XML file ready for testing

## Configuration Setup

### Step 1: Access Configuration Page

Navigate to Configuration from the sidebar to set up your AI model credentials.

### Step 2: Azure OpenAI Configuration

You are required to get either a personal sandbox subscription or have a BDP workspace to access the Azure OpenAI LLMs.

#### Configuration Process

1. Enter Credentials:

* Paste your Azure OpenAI Key in the password field
* Enter your endpoint URL
* The system auto-configures API version (2025-01-01-preview) and model (gpt-4o). The API version will change as new versions are released by OpenAI and available.

1. Save Configuration:

* Click " Save Configuration"
* Wait for the success confirmation
* Balloons animation indicates successful save

1. Verify Setup:

* Check the Configuration Status section
* Green checkmark () indicates complete configuration
* Warning symbol (⚠️) indicates missing information

### Step 3: Test Connection

1. Run Connection Test:

* Click "🧪 Test Connection" button
* Wait for the test to complete
* Review the test response

1. Troubleshooting Test Issues:

* Authentication Error: Verify API key is correct
* Resource Not Found: Check endpoint URL and model deployment
* Quota/Rate Limit: Check Azure resource limits
* Timeout: Check network connectivity

### Configuration Files

The system automatically creates/updates:

* .env file with encrypted credentials
* Configuration backup files for recovery

## Discovery Module

The Discovery module is the core feature for extracting patterns from XML files using AI assistance.

### Overview

The Discovery page provides five main tabs:

* Extract: Upload and extract patterns from XML
* Verify: Validate and confirm extracted patterns
* Save: Store patterns in databases
* Manage Custom Patterns: Handle personal workspace patterns
* Manage Shared Patterns: Handle team shared patterns

### Tab 1: Extract Patterns

#### Step-by-Step Process

1. Upload XML File:

* Click the file uploader in the Extract tab
* Select your XML file (supported: .xml files)
* File metrics will display (name, size, status)

1. Select API Context:

* Choose the relevant API from dropdown:
* LATAM: LATAM Airlines API
* LH: Lufthansa API
* LHG: Lufthansa Group API
* AFKL: Air France-KLM API

1. Choose API Version:

* Select from available versions in dropdown
* Or manually enter version (e.g., "17.2", "18.2")
* Version helps context-specific pattern extraction

1. Initiate Extraction:

* Click "🚀 Start Genie Pattern Extraction"
* Monitor the progress status
* AI processes the XML structure and identifies patterns

#### Understanding Extraction Results

Pattern Information Display:

Each extracted pattern shows:

* Pattern Name: Descriptive name generated by AI
* XPath: Location path in XML structure
* Description: AI-generated explanation of the pattern
* Category: Auto-classified type (flight, passenger, fare, etc.)
* Example Data: Sample values from the XML

Pattern Categories:

* Flight: Flight segments, routes, scheduling
* Passenger: Traveler details, personal information
* Fare: Pricing, costs, fare rules
* Booking: Reservations, PNR, confirmation codes
* Airline: Carrier information, airline codes
* Custom: User-defined or unique patterns

#### Extraction Tips

* File Size: Optimal XML files are 10KB-1MB
* Structure: Well-formed XML produces better results
* Content: Rich data yields more meaningful patterns
* API Context: Correct API selection improves accuracy

#### Interactive Pattern Analysis

Genie Chatbot:

After pattern extraction, use the " Ask Genie About Extraction" button to:

1. Extraction Summary: Get detailed summaries of extracted patterns

* Total patterns found with descriptions
* Pattern categories and types identified
* Quality assessment of extraction results

1. Pattern Quality Analysis: Understand extraction effectiveness

* Coverage assessment of XML elements
* Pattern completeness evaluation
* Recommendations for improvement

1. Next Steps Guidance: Get recommendations for:

* Pattern verification strategies
* Saving and organization approaches
* Additional extraction opportunities

Quick Actions:

* Extraction Summary: "Can you summarize the pattern extraction results?"
* Pattern Quality: "How good are the extracted patterns?"
* Next Steps: "What should I do next with these patterns?"

### Tab 2: Verify Patterns

#### Verification Process

1. Review Extracted Patterns:

* All extracted patterns appear in expandable sections
* Each pattern shows detailed information
* Status indicates "Verified" or "Unverified"

1. Individual Pattern Verification:

* Click " Verify Pattern" for each pattern
* AI analyzes the pattern against XML context
* Verification adds quality assurance

1. Batch Verification:

* Use " Verify All Patterns" for bulk processing
* Faster for multiple patterns
* Each pattern still gets individual AI review

1. Understanding Verification Results:

* Verified: Pattern confirmed by AI as accurate
* ⏳ Unverified: Pattern extracted but not validated
* ❌ Failed: Pattern couldn't be verified (rare)

#### Verification Benefits

* Quality Assurance: Ensures pattern accuracy
* Confidence: Verified patterns are more reliable
* Documentation: Adds verification metadata
* Filtering: Verified patterns can be filtered in searches

### Tab 3: Save Patterns

#### Save to Personal Workspace

1. Pattern Selection:

* Review all available patterns
* Check boxes to select patterns for saving
* Use "Select All" or "Select Verified Only" for bulk selection

1. API Configuration:

* Select target API from dropdown
* Choose or enter API version
* Configuration determines storage location

1. Validation and Save:

* Review the save summary (patterns count, verified count)
* Click " Save X Selected Patterns"
* Monitor the save progress status

#### Save to Shared Workspace

1. Pattern Selection:

* Choose patterns to share with team
* Consider sharing only verified patterns
* Select patterns valuable for team collaboration

1. Category Assignment:

* Choose from predefined categories:
* Flight: Flight-related patterns
* Passenger: Passenger information patterns
* Fare: Pricing and fare patterns
* Booking: Reservation patterns
* Airline: Carrier information patterns
* Or create custom category

1. API Configuration for Shared Patterns:

* Select API: Choose the relevant API (LATAM, LH, LHG, AFKL)
* Select API Version: Choose or enter the API version
* This metadata helps team members understand pattern context

1. Save Process:

* Review save summary including API information
* Click "🌐 Save X Patterns to Shared Workspace"
* Patterns become available to all team members

#### Storage Locations

* Personal Workspace: Workspace-specific database
* Shared Workspace: Global shared database with API/version metadata
* Backup: Automatic JSON exports available

### Tab 4: Manage Custom Patterns

#### Overview

Manage patterns saved in your personal workspace database.

#### Features

1. View Saved Patterns:

* List all patterns in current workspace
* Display pattern details and metadata
* Show save date and verification status

1. Pattern Management:

* Edit Categories: Change pattern classifications
* Delete Patterns: Remove unwanted patterns
* Export Data: Download patterns as CSV/JSON

1. Filtering and Search:

* Filter by category, API, or verification status
* Search by pattern name or description
* Sort by various criteria

### Tab 5: Manage Shared Patterns

#### Overview

Handle patterns in the shared team workspace.

#### Team Collaboration Features

1. View Shared Patterns:

* Access patterns saved by team members
* See API and version information
* Review pattern metadata and descriptions

1. Pattern Management:

* Delete Shared Patterns: Remove patterns (admin access)
* Export Shared Library: Download team patterns
* Import Patterns: Add patterns from external sources

1. Quality Control:

* Review team contributions
* Maintain pattern quality standards
* Organize shared library structure

## Pattern Identification

### Overview

The Identify module helps you analyze unknown XML files by comparing them against your saved patterns.

### How It Works

1. Pattern Matching: Compares uploaded XML against saved patterns
2. AI Analysis: Uses GPT-4o to verify pattern matches
3. Confidence Scoring: Provides match confidence levels
4. Detailed Results: Shows which patterns match and why

### Step-by-Step Process

#### Step 1: Upload XML for Analysis

1. Select Workspace:

* Choose the workspace containing relevant patterns
* Workspace determines which patterns are used for comparison

1. Upload XML File:

* Click the file uploader
* Select your unknown XML file
* File metrics display (name, size, analysis status)

1. Pre-Analysis Check:

* System verifies pattern availability
* Shows count of workspace and shared patterns
* Warns if no patterns available for comparison

#### Step 2: Run Pattern Analysis

1. Start Analysis:

* Click "Start Pattern Analysis"
* AI processes XML against all available patterns
* Progress status shows analysis phases

1. Analysis Process:

* Structure Analysis: XML structure parsing
* Pattern Matching: Compare against saved patterns
* AI Verification: GPT-4o validates matches
* Result Compilation: Generate analysis report

#### Step 2.5: Apply Filters (Optional)

Pattern Selection Filters:

Before running analysis, you can optionally filter which patterns to test against:

1. Airline Filter:

* Select specific airlines from dropdown (e.g., "American Airlines", "Delta")
* Choose "All Airlines" to test all available patterns
* Reduces analysis time by focusing on relevant patterns

1. API Version Filter:

* Select specific API versions (e.g., "v1.0", "v2.1")
* Choose "All Versions" to test all available versions
* Helps target specific API implementations

1. Filter Summary:

* Active filters are displayed with clear indicators
* Shows exactly which patterns will be tested
* Can combine airline and version filters for precise targeting

#### Step 3: Review Results

Analysis Table:

* Airline: Actual airline name (e.g., "American Airlines") or API source
* API Version: Specific version information (e.g., "v2.1", "18.2")
* Section: XML section or XPath where pattern was found
* Validation Rule: Description of what was matched
* Verified: Yes/No indicating if pattern matched
* Reason: AI explanation of match/non-match

Matched Airlines Display:

* Shows airlines with confirmed pattern matches
* Format: "Airline+Version" (e.g., "AmericanAirlinesv2.1")
* Highlighted in yellow for visibility

Interactive Chatbot:

After analysis completes, use the " Ask Genie About Results" button to:

* Ask questions about the analysis results
* Get summaries of matched patterns
* Understand data quality assessments
* Receive recommendations for next steps

#### Understanding Results

Verification Statuses:

* Yes: Pattern definitively matches XML structure
* No: Pattern doesn't match this XML

Common Reasons for Matches:

* Element structure similarity
* Data format consistency
* Naming convention matches
* Value pattern recognition

Common Reasons for Non-Matches:

* Missing XML elements
* Different data formats
* Structural differences
* Namespace variations

### Pattern Library

#### Overview

The Pattern Library provides a unified view of all your saved patterns.

#### Features

1. Comprehensive View: Shows patterns from both workspace and shared libraries
2. Advanced Filtering: Filter by source, category, and airline
3. Export Options: Download filtered results as CSV
4. Real-time Updates: Refresh to see latest patterns

#### Filtering Options

* Source Filter:
* Shared Workspace: Team patterns with API/version info
* Custom (API): Personal workspace patterns by API
* Category Filter:
* Flight, Passenger, Fare, Booking, Airline
* Custom categories created by users
* Airline Filter:
* Filter by specific airlines or "Shared"
* Shows actual API names for shared patterns

#### Pattern Information Display

Each pattern shows:

* Source: Origin (Shared Workspace or Custom)
* Name: Pattern identifier
* Category: Classification
* Airline: API name or "Shared"
* API Version: Version info or "N/A"
* XPath: Location in XML structure
* Description: Pattern explanation

## Workspace Management

### Overview

Workspaces organize your patterns by project, use case, or team structure.

### Workspace Selection

* Sidebar Selector: Choose active workspace from " Workspace Selection"
* Blank Default: No workspace selected on startup
* Context Switching: Easy switching between projects
* Enhanced Management: Color-coded Add (green) and Delete (red) workspace buttons
* Import/Export: Pattern import/export functionality in Discovery page management tabs

### Creating Workspaces

Workspaces are created automatically when:

1. Saving patterns with new API/project combination
2. Using the UseCase Manager for new projects
3. Database initialization for new contexts

### Workspace Structure

Each workspace contains:

* Pattern Database: SQLite database with patterns
* API Configuration: Supported APIs and versions
* Section Mappings: XPath to pattern relationships
* Metadata: Creation date, last modified, pattern counts

### Best Practices

1. Naming Convention: Use descriptive workspace names
2. Organization: Separate by project or API type
3. Cleanup: Regular maintenance of unused workspaces
4. Backup: Export important workspace patterns

## API Management

### Supported APIs

The system supports four airline APIs:

1. LATAM: LATAM Airlines API
2. LH: Lufthansa API
3. LHG: Lufthansa Group API
4. AFKL: Air France-KLM API

### API Configuration

#### Adding New APIs

1. Access Management:

* Go to Discovery > Save tab
* Expand "⚙️ Manage APIs" section

1. Add API:

* Enter valid API name (LATAM, LH, LHG, AFKL only)
* Click "➕ Add API"
* System validates name against allowed list

1. Validation:

* Only approved API names accepted
* Workspace names like "LATAM\_OVRS" rejected
* Error messages guide correct input

#### Managing API Versions

1. Automatic Addition:

* Versions added when saving patterns
* Manual entry supported if no versions exist

1. Version Management:

* View existing versions in dropdowns
* Add new versions through pattern saving
* Version format: "X.Y" (e.g., "17.2", "18.2")

#### API Deletion

1. Safety Checks:

* Cannot delete APIs with associated patterns
* Warning messages prevent data loss

1. Deletion Process:

* Select API to delete
* Click "🗑️ Delete"
* Confirm deletion if no patterns exist

### API-Pattern Relationships

* One-to-Many: Each API can have multiple versions
* Version-Pattern: Patterns linked to specific API versions
* Shared Patterns: Include API and version metadata
* Cross-Reference: Patterns can reference multiple APIs

## Advanced Features

### Pattern Categories

#### Predefined Categories

* Flight: Segments, routes, schedules, flight numbers
* Passenger: Names, contact info, preferences, special services
* Fare: Pricing, taxes, fare basis, restrictions
* Booking: PNR, confirmations, reservations, status
* Airline: Carrier codes, names, operating airlines
* User Created: Custom patterns by users

#### Custom Categories

1. Creation: Use "custom" option when saving to shared workspace
2. Naming: Enter descriptive category name
3. Organization: Custom categories appear in all filters
4. Management: Edit and organize through pattern management

### Data Export and Import

#### Export Options

1. CSV Export:

* Pattern Library: Export filtered patterns as CSV
* Full data: All pattern information included
* Use case: Data analysis, reporting, backup

1. JSON Export:

* Raw pattern data in JSON format
* Complete pattern structure preserved
* Use case: Data migration, integration, backup

#### Export Process

1. Filter Data: Apply desired filters in Pattern Library
2. Export Action: Click "📄 Export Saved Patterns"
3. Download: Click "⬇️ Download CSV" button
4. File Location: Check browser downloads folder

### Pattern Verification Engine

#### AI-Powered Verification

* GPT-4o Analysis: Each pattern analyzed by AI
* Context Awareness: XML context considered in verification
* Confidence Scoring: Internal confidence metrics
* Quality Assurance: Multiple validation layers

#### Verification Criteria

* Structural Validity: XPath accuracy
* Data Consistency: Value format verification
* Context Relevance: Pattern relevance to XML content
* Completeness: Pattern coverage adequacy

### Database Architecture

#### Storage Structure

* SQLite Databases: Local storage for reliability
* Multi-Database: Separate workspace and shared databases
* Relational Design: Normalized structure for efficiency
* Indexing: Optimized for pattern queries

#### Database Tables

1. api: API definitions and metadata
2. apiversion: Version tracking per API
3. api\_section: XML sections and paths
4. pattern\_details: Pattern information and prompts
5. section\_pattern\_mapping: Relationships between sections and patterns
6. default\_patterns: Shared workspace patterns with API metadata

### Session Management

* State Persistence: Patterns retained during session
* Cross-Page Data: Patterns available across all pages
* Memory Management: Efficient data handling
* Session Recovery: Automatic state restoration

## Troubleshooting

### Common Issues and Solutions

#### Configuration Problems

Issue: "Configuration Incomplete" Warning

* Cause: Missing Azure OpenAI credentials
* Solution:

1. Go to Configuration page
2. Enter all required fields (Key, Endpoint)
3. Save configuration
4. Test connection

Issue: "Test Connection Failed"

* Cause: Invalid credentials or connectivity
* Solution:

1. Verify API key is active and correct
2. Check endpoint URL format
3. Ensure Azure resource is running
4. Check network connectivity

#### Pattern Extraction Issues

Issue: "No Patterns Extracted"

* Cause: XML file issues or AI processing errors
* Solution:

1. Verify XML file is valid and well-formed
2. Check file size (should be reasonable, not too large)
3. Ensure API selection matches XML content
4. Try different XML file for testing

Issue: "Extraction Takes Too Long"

* Cause: Large XML files or API rate limits
* Solution:

1. Use smaller XML files (< 1MB recommended)
2. Check Azure OpenAI quota limits
3. Wait for current request to complete
4. Retry after few minutes

#### Pattern Identification Issues

Issue: "No Validation Rules Found"

* Cause: No saved patterns in workspace or shared library
* Solution:

1. Save patterns from Discovery page first
2. Check if correct workspace is selected
3. Verify patterns exist in Pattern Library
4. Use different workspace with patterns

Issue: "Analysis Shows No Matches"

* Cause: XML doesn't match existing patterns
* Solution:

1. Verify XML is from expected API
2. Check if patterns are from same API version
3. Ensure XML structure is complete
4. Try with different XML file

#### Database and Storage Issues

Issue: "Failed to Save Patterns"

* Cause: Database connection or permission issues
* Solution:

1. Check disk space availability
2. Verify write permissions to database directory
3. Restart application
4. Check for database corruption

Issue: "Patterns Disappear When Switching Pages"

* Cause: Session state management issues
* Solution:

1. Complete pattern saving before page switching
2. Use browser refresh if needed
3. Re-extract patterns if lost
4. Save patterns immediately after extraction

#### Performance Issues

Issue: "Application Runs Slowly"

* Cause: Large datasets or resource constraints
* Solution:

1. Clear browser cache
2. Restart Streamlit application
3. Check system memory usage
4. Use smaller datasets for testing

### Error Messages and Meanings

#### Configuration Errors

* "Authentication Error": Invalid API key
* "Resource Not Found": Incorrect endpoint or model name
* "Quota/Rate Limit": Azure resource limits exceeded
* "Timeout Error": Network connectivity issues

#### Pattern Processing Errors

* "Invalid XML Structure": Malformed XML file
* "Processing Failed": AI analysis error
* "Pattern Not Found": XPath resolution failed
* "Verification Failed": Pattern validation error

### Getting Help

#### Debug Information

When reporting issues, include:

1. Error Messages: Complete error text
2. File Information: XML file size and source
3. Configuration: API and version being used
4. Steps: What you were doing when error occurred
5. Browser: Browser type and version

#### Log Files

Application logs contain detailed information:

* Location: Check console output in browser
* Content: Error messages, API calls, processing steps
* Usage: Share relevant log sections when requesting help

## Best Practices

### File Management

#### XML File Preparation

1. Validation: Ensure XML is well-formed before upload
2. Size Optimization: Keep files under 1MB for best performance
3. Content Quality: Use files with rich, representative data
4. Naming: Use descriptive filenames for easy identification

#### Organization

1. File Structure: Organize XML files by API and version
2. Backup: Keep original files after pattern extraction
3. Version Control: Track XML file changes over time
4. Documentation: Document file sources and contexts

### Pattern Management

#### Quality Assurance

1. Verification: Always verify patterns before saving
2. Review: Manually review AI-generated descriptions
3. Testing: Test patterns with multiple XML files
4. Cleanup: Remove outdated or incorrect patterns

#### Organization Strategies

1. Naming: Use consistent pattern naming conventions
2. Categories: Properly categorize patterns for easy filtering
3. Documentation: Add detailed descriptions to patterns
4. Versioning: Track pattern changes and improvements

### Workspace Organization

#### Structure Planning

1. Purpose-Based: Create workspaces by project or use case
2. API-Based: Separate workspaces by airline API
3. Team-Based: Organize for team collaboration needs
4. Environment-Based: Separate development, testing, production

#### Maintenance

1. Regular Cleanup: Remove unused patterns and workspaces
2. Backup: Export important workspace data regularly
3. Documentation: Maintain workspace purpose documentation
4. Access Control: Manage team access to shared workspaces

### Performance Optimization

#### Efficient Usage

1. Batch Processing: Process multiple patterns together
2. Optimal Timing: Use system during off-peak hours
3. Resource Management: Monitor Azure OpenAI usage
4. Caching: Leverage browser caching for better performance

#### Cost Management

1. Token Awareness: Understand Azure OpenAI token costs
2. Efficient Prompts: Use appropriate prompt engineering
3. Batch Operations: Combine API calls when possible
4. Monitoring: Track API usage and costs regularly

### Security Practices

#### Credential Management

1. Secure Storage: Never share Azure OpenAI credentials
2. Regular Rotation: Update API keys periodically
3. Access Control: Limit credential access to authorized users
4. Monitoring: Watch for unusual API usage patterns

#### Data Protection

1. Sensitive Data: Avoid uploading sensitive XML content
2. Local Storage: Understand data is stored locally
3. Backup Security: Secure pattern export files
4. Team Access: Control shared workspace access

## FAQ

### General Questions

Q: What is AssistedDiscovery?

A: AssistedDiscovery is an AI-powered XML pattern analysis tool that helps developers understand, extract, and manage patterns from airline API XML responses using GPT-4o.

Q: Do I need programming knowledge to use this tool?

A: No, the tool is designed with a user-friendly interface. Basic understanding of XML structure is helpful but not required.

Q: Can I use this tool offline?

A: The tool requires internet connection for AI processing (Azure OpenAI API calls), but pattern storage and management work locally.

### Configuration Questions

Q: What Azure OpenAI subscription do I need?

A: You need an active Azure OpenAI subscription with GPT-4o model deployment. Standard pay-as-you-go pricing applies.

Q: Can I use regular OpenAI instead of Azure OpenAI?

A: Currently, the tool is configured specifically for Azure OpenAI. Regular OpenAI API is not supported.

Q: How much does it cost to run pattern analysis?

A: Costs depend on Azure OpenAI usage. Typical pattern extraction costs $0.01-0.10 per XML file, depending on size and complexity.

### Pattern Questions

Q: What types of XML files work best?

A: Well-formed XML files from airline APIs work best. Files should be 10KB-1MB in size with representative data structures.

Q: Can I extract patterns from non-airline XML?

A: The tool is optimized for airline APIs but can work with any XML. Results may vary for non-airline content.

Q: How accurate is the AI pattern extraction?

A: GPT-4o provides highly accurate pattern extraction. Always verify patterns before saving for best results.

Q: Can I edit extracted patterns?

A: Currently, patterns cannot be edited after extraction. Re-extract with different prompts or contexts if changes are needed.

### Workspace Questions

Q: How many workspaces can I create?

A: There's no hard limit on workspaces. Performance may be affected with very large numbers of workspaces.

Q: Can I share workspaces with team members?

A: Workspaces are local to each installation. Use the shared workspace feature for team collaboration.

Q: What happens if I delete a workspace?

A: Workspace deletion removes all associated patterns and data. Always export important data before deletion.

### Technical Questions

Q: Where are my patterns stored?

A: Patterns are stored in local SQLite databases in the application directory. Workspace patterns and shared patterns use separate databases.

Q: Can I backup my patterns?

A: Yes, use the export features to backup patterns as CSV or JSON files. Database files can also be backed up directly.

Q: What happens if the application crashes?

A: Session data may be lost, but saved patterns in databases are preserved. Re-extract patterns if session data is lost.

Q: Can I run multiple instances of the application?

A: Yes, but they share the same database files. Be careful with concurrent modifications.

### Integration Questions

Q: Can I integrate this with other tools?

A: Pattern data can be exported as CSV/JSON for integration. Direct API integration is not currently available.

Q: Can I automate pattern extraction?

A: The current version requires manual operation through the web interface. Automation features may be added in future versions.

Q: Can I import patterns from other sources?

A: Patterns can be imported through the shared workspace management features, though manual formatting may be required.

### Troubleshooting Questions

Q: Why are my patterns showing random dates?

A: This was a database column mapping issue that has been fixed. Restart the application to see correct API and version information.

Q: Why don't I see extracted patterns on other pages?

A: Patterns must be saved to databases to appear on other pages. Session patterns only exist during extraction.

Q: Why does pattern identification show "No validation rules found"?

A: This means no patterns are available for comparison. Save patterns from the Discovery page first.

### Support and Updates

Q: How do I get support?

A: Check this user guide first, then review troubleshooting section. For technical issues, check application logs and error messages.

Q: How often is the tool updated?

A: Updates are released as needed for bug fixes and feature enhancements. Check the repository for latest versions.

Q: Can I contribute to the project?

A: Contribution guidelines depend on the project's open-source status. Check the repository for contribution information.

## Conclusion

AssistedDiscovery provides a powerful, AI-driven approach to XML pattern analysis specifically designed for airline API development and analysis. By following this comprehensive guide, users can effectively:

* Set up and configure the system for their needs
* Extract meaningful patterns from complex XML structures
* Organize and manage patterns efficiently
* Collaborate with teams through shared workspaces
* Identify patterns in unknown XML files
* Maintain high-quality pattern libraries

### Key Success Factors

1. Proper Configuration: Ensure Azure OpenAI setup is correct
2. Quality Data: Use representative XML files for pattern extraction
3. Verification: Always verify patterns before saving
4. Organization: Maintain clean workspace and pattern organization
5. Documentation: Document patterns and workspaces for team use

### Getting the Most Value

* Start Small: Begin with a few XML files to understand the system
* Build Gradually: Add more patterns and workspaces over time
* Team Collaboration: Use shared workspaces for team efficiency
* Regular Maintenance: Keep patterns and workspaces organized
* Continuous Learning: Explore advanced features as you become comfortable

For additional support or questions not covered in this guide, refer to the application's built-in help features and error messages, which provide context-specific guidance for troubleshooting and optimization.

This user guide covers version 1.0 of AssistedDiscovery. Features and interfaces may evolve in future releases.