# AssistedDiscovery v1.0 - Complete User Guide

## Table of Contents

1. [Introduction](#introduction)
2. [Getting Started](#getting-started)
3. [Configuration](#configuration)
4. [Core Workflows](#core-workflows)
5. [Node Configuration](#node-configuration)
6. [Discovery Workflow](#discovery-workflow)
7. [Pattern Manager](#pattern-manager)
8. [Identify Workflow](#identify-workflow)
9. [Workspaces](#workspaces)
10. [Troubleshooting](#troubleshooting)
11. [Best Practices](#best-practices)

## Introduction

### What is AssistedDiscovery?

AssistedDiscovery is an AI-powered tool that helps you understand and document NDC (New Distribution Capability) XML message structures. It:

* **Extracts node structures** from XML files automatically
* **Discovers relationships** between nodes (who references whom)
* **Generates reusable patterns** for future validation
* **Identifies changes** in XML structure over time
* **Validates XML files** against known patterns

### Key Concepts

**Pattern**: A reusable template describing how a specific node type should look (its structure, attributes, children)

**Relationship**: A reference from one node to another (e.g., Passenger → Segment)

**Discovery**: The process of analyzing XML structure and generating reusable patterns

**Identify**: The process of comparing XML against known patterns

**Workspace**: An isolated environment for a specific airline or project

### Important: Understanding AI-Powered Analysis

AssistedDiscovery uses **Large Language Models (LLMs)** to analyze XML structure. This means:

**⚠️ Results May Vary**: - **LLMs can make mistakes**: Like humans, AI can misinterpret or miss information - **Non-deterministic**: Running Discovery/Identify on the same XML file multiple times may produce slightly different results - **Confidence scores matter**: Always review low-confidence matches (< 85%) - **Human validation required**: Treat results as AI-assisted suggestions, not absolute truth

**Typical Accuracy**: - **Node extraction**: 90-95% accurate - **Relationship discovery**: 85-90% accurate - **Pattern matching**: 90-95% accurate

**Best Practices**: 1. **Always review results**: Don’t trust blindly 2. **Validate unexpected discoveries**: AI might find real issues OR make mistakes 3. **Check low-confidence matches**: < 85% confidence needs human verification 4. **Run multiple times if uncertain**: Compare results for consistency 5. **Report issues**: Help improve the system by reporting errors

**We Need Your Feedback!**: Your feedback helps improve AssistedDiscovery. Please report: - **False positives**: AI found relationships that don’t exist - **False negatives**: AI missed relationships that do exist - **Incorrect patterns**: AI generated wrong patterns - **Inconsistent results**: Different results on same XML - **Quality issues**: Any accuracy or reliability problems

**How to Report**: - **Team Contact**: Reach out to nikhilkrishna.lepakshi@amadeus.com - **Include**: Error message, Run ID, log files, sample XML (if possible)

## Getting Started

### Installation

#### Portable Distribution (Recommended for Users)

1. **Extract the ZIP file**:

* unzip AssistedDiscovery-Portable-\*.zip  
  cd AssistedDiscovery-Portable-\*

1. **Run setup** (one-time only):

* # macOS/Linux  
  ./setup.sh  
    
  # Windows  
  setup.bat

1. **Start the application**:

* # macOS/Linux  
  ./start\_app.sh  
    
  # Windows  
  start\_app.bat

1. **Access the UI**:
   * Open your browser
   * Go to: http://localhost:8501

### First-Time Setup Checklist

* Application installed and running
* Browser opens to http://localhost:8501
* LLM credentials configured (see [Configuration](#configuration))
* Test workspace created (see [Quick Start: Create a Workspace](#quick-start-create-a-workspace))
* Sample XML file ready for testing

### Quick Start: Create a Workspace

Before running Discovery, you should create a **workspace** for your project or airline.

**What is a Workspace?** A workspace is an isolated environment that stores all your data (patterns, runs, configurations) separately. Think of it as a project folder.

**Why use workspaces?** - ✅ Separate data per airline (LATAM vs United vs Delta) - ✅ Keep test data separate from production - ✅ Clean organization and comparison

**How to Create a Workspace:**

1. **Access Config Page**:
   * Click **⚙️ Config** in the sidebar
2. **Scroll to Workspace Management**:
   * Find the **📁 Workspace Management** section
3. **Add New Workspace**:
   * Enter workspace name in the text box
   * Example names: WestJet, LATAM, Testing, Production
   * Use alphanumeric characters only (no spaces)
4. **Click ➕ Add Workspace**:

* ✅ Workspace 'WestJet' created successfully!

1. **Switch to Your Workspace**:
   * At the top of the sidebar, you’ll see **Workspace: [dropdown]**
   * Select your newly created workspace
   * All operations now use this workspace

**Example Setup:**

Workspaces:  
├─ default (Built-in, always available)  
├─ WestJet (For WestJet Airlines)  
├─ LATAM (For LATAM Airlines)  
└─ Testing (For experiments)

**Important**: Always check which workspace you’re in before running Discovery or Identify!

For detailed workspace management, see the [Workspaces](#workspaces) section.

## Configuration

### LLM Configuration

AssistedDiscovery uses AI (Large Language Model) to extract information from XML. You need to configure your LLM provider.

#### Step 1: Access Configuration

1. Click **⚙️ Config** in the sidebar
2. Scroll to **🤖 LLM Configuration** section

#### Step 2: Select Provider

**Supported Providers**: - **Azure OpenAI**

#### Step 3: Configure Azure OpenAI

**Required Information**: - **Endpoint**: Your Azure OpenAI endpoint URL - Format: https://your-resource.openai.azure.com/ - **API Key**: Your Azure OpenAI API key - Found in Azure Portal → OpenAI Resource → Keys - **API Version**: API version (default: 2025-01-01-preview) - **Model Deployment**: Your deployment name (e.g., gpt-4o)

**Configuration Steps**:

1. Select **azure** from LLM Provider dropdown
2. Enter your **Azure OpenAI Endpoint**
3. Enter your **API Key**
4. Verify **API Version** (usually default is correct)
5. Enter your **Model Deployment Name**
6. Configure common settings:
   * **Max Tokens**: 4000 (default, adjust if needed)
   * **Temperature**: 0.1 (low for consistency)
   * **Top P**: 0.0 (deterministic)
7. Click **💾 Save Configuration**
8. Click **🔍 Test Connection** to verify

**Expected Result**:

✅ Connection successful!  
Provider: azure

#### Step 4: Apply Configuration

After saving, you’ll see:

✅ Configuration saved!  
⚠️ Please restart the backend for changes to take effect.

To apply your configuration, restart the entire application:

1. **Stop the application**:
   * Go to the terminal window where the app is running
   * Press Ctrl+C (or Cmd+C on Mac)
   * Wait for the app to fully stop
2. **Start the application again**:

* # macOS/Linux  
  ./start\_app.sh  
    
  # Windows  
  start\_app.bat

1. **Verify the configuration**:
   * Go back to **⚙️ Config** → **🤖 LLM Configuration**
   * Your settings should be preserved
   * Click **🔍 Test Connection** to verify

### Log File Location

Application logs are stored in platform-specific locations:

* **macOS**: ~/Library/Logs/AssistedDiscovery/assisted\_discovery.log
* **Windows**: %LOCALAPPDATA%\AssistedDiscovery\Logs\assisted\_discovery.log

**To view logs via UI**: 1. Go to **⚙️ Config** page 2. Scroll to **📋 Application Logs** section 3. Click **📂 Open Log Folder** button

### Managing Workspaces

Access workspace management in **⚙️ Config** page.

#### Create New Workspace

1. Go to **⚙️ Config**
2. Scroll to **📁 Workspace Management**
3. Enter workspace name:

* New Workspace: WestJet
  + Use airline code or project name
  + Alphanumeric only (no spaces)
  + Examples: WestJet, United, Test, Production

1. Click **➕ Add Workspace**
2. **Result**:

* ✅ Workspace 'WestJet' created successfully!

#### Switch Workspace

**In Sidebar**:

Workspace: [WestJet ▼]

**Effect**: All pages now show data from selected workspace.

#### Delete Workspace

**⚠️ Warning**: This permanently deletes ALL data in workspace!

1. Go to **⚙️ Config** → **Delete Workspace**
2. Select workspace to delete:

* Workspace to delete: [WestJet ▼]
  + Cannot delete “default” workspace
  + Must have at least one workspace

1. **Warning message**:

* ⚠️ This will permanently delete the workspace and all its data  
  (patterns, runs, node facts)!

1. Click **🗑️ Delete Workspace**
2. Confirmation:

* ✅ Workspace 'WestJet' and its database deleted.

**What gets deleted**: - All Patterns - All Relationships - All Node Configurations - All Discovery/Identify runs - Workspace database file (.db file deleted from disk)

## Core Workflows

AssistedDiscovery has four main workflows:

### 1. Discovery Workflow 🔍

**Purpose**: Analyze existing airline XML files to extract and generate reusable patterns

**When to use**: - Analyzing an existing airline’s XML format - Creating pattern library for known/existing airlines - Need to understand and document XML structure - Generating patterns for future validation

**Output**: - Patterns (reusable templates for node structures) - Relationships (node references)

**Typical Duration**: 2-5 minutes for standard XML

### 2. Identify Workflow 🎯

**Purpose**: Validate XML from new/unknown airlines against saved patterns from existing airlines

**When to use**: - After Discovery has generated patterns from existing airlines - Validating new airline XML files against known patterns - Checking how closely new airline matches existing patterns - Identifying deviations and differences from standard patterns

**Output**: - Pattern matches (how closely new XML matches saved patterns) - Confidence scores - Deviation reports

**Typical Duration**: 30 seconds for standard XML

### 3. Pattern Management 🎨

**Purpose**: View, export, and manage discovered patterns

**When to use**: - Review extracted patterns - Export patterns for documentation - Import patterns from another workspace

**Output**: - Pattern library - Exportable JSON files

### 4. Node Configuration 📋

**Purpose**: Configure which nodes to extract and their expected references

**When to use**: - Before Discovery

**Output**: - Node Configuration rules - Expected reference definitions

## Node Configuration

Node Configuration tells AssistedDiscovery **which nodes to extract** and **what references to expect**.

### Why Configure Nodes?

Node configuration is **required** before running Discovery. You must configure which nodes to extract from your XML.

**Benefits**: - ✅ **Controlled Extraction**: Only extract nodes you care about - ✅ **Better Relationships**: Define expected references between nodes - ✅ **Consistent Results**: Same extraction rules across runs - ✅ **Required Step**: Discovery will only extract nodes that are configured and enabled

### Access Node Configuration

1. Click **📋 Node Config** in sidebar
2. You’ll see two tabs:
   * **📤 Analyze XML**: Upload XML to see available nodes
   * **⚙️ Manage Configurations**: Create/edit node configs

### Tab 1: Analyze XML

**Purpose**: Discover which nodes exist in your XML and configure them using an interactive tree view.

#### Steps:

1. **Upload XML File**:
   * Click **Choose XML file** or drag-and-drop
   * Select your NDC XML file
   * System analyzes the XML structure automatically
2. **Review Detected Information**:

* ✅ Discovered 47 nodes in 19.2/OrderViewRS - Airline: WS  
    
  Spec Version: 19.2  
  Airline: WS (WestJet)  
  Total Nodes: 47  
  Configured: 9

1. **Understanding Node Selection**:

* The tree view shows your XML structure hierarchically:
* **How Node Selection Works:**
  + **Check a parent node** → Extracts that parent + all its descendants
  + **Check a child only** → Extracts that child + its descendants (parent NOT extracted)
  + **Check a leaf node** → Extracts only that specific node
* Each checked node becomes a “root” for hierarchical extraction. Only explicitly checked nodes are saved.

1. **Select Nodes for Extraction**:

* **Tree View** (Left Panel):
* 🌳 Node Hierarchy  
    
  ▶ IATA\_AirShoppingRS  
   ▶ PayloadAttributes  
   ▼ Response  
   ▼ DataLists  
   ✓ DatedMarketingSegmentList  
   ✓ DatedOperatingLegList  
   ✓ DatedOperatingSegmentList  
   ✓ DisclosureList  
   ✓ OriginDestList
  + Click ▶ to expand/collapse nodes
  + Check ✓ boxes to enable extraction
  + Parent selection auto-enables descendants (shown with ✓ but not saved individually)
* **Node Properties** (Right Panel):
* ✅ 9 parent nodes selected for extraction  
  🔁 82 descendants will be auto-extracted (shown checked in tree)  
    
  Selected nodes for extraction:  
  • DatedMarketingSegmentList  
  • DatedOperatingLegList  
  • DatedOperatingSegmentList  
  ... and 6 more

1. **Save Configuration**:
   * Click **💾 Save All Configurations**
   * Result:
   * ✅ Saved 47 node configurations!  
      • 9 parent nodes enabled for extraction  
     💡 Configurations saved successfully. Parent nodes will auto-extract  
      their descendants during Discovery.

#### Important Notes:

* **Tree starts collapsed** when you reload - manually expand to see nodes
* **Checkmarks persist** across page reloads
* **Only parent nodes are saved** - descendants are auto-extracted
* **No descendants expanded automatically** - you control the tree view

#### Disabling Nodes:

To **disable** a node that was previously enabled for extraction:

1. **Upload the same XML file** in the Analyze XML tab
2. **Expand the tree** to find the enabled node (shown with ✓)
3. **Uncheck the node** by clicking on its checkbox
4. **Click Save All Configurations**

**What happens when you disable a node:** - The node is marked as enabled = false in the database - The node will **NOT be extracted** during Discovery - All descendants of that node will also be skipped during extraction - The configuration remains saved but inactive

**Example:**

Before: ✓ PassengerList (enabled)  
After: ☐ PassengerList (disabled)

When you save, you’ll see:

✅ Saved 47 node configurations!  
 • 8 parent nodes enabled for extraction  
 • 1 parent node disabled (not extracted)

**Important**: Disabling a parent node automatically disables all its descendants. They will not be extracted during Discovery, even if they were previously enabled.

#### Re-loading Your Configuration:

When you upload the same XML file later: - Previously selected nodes will show with checkmarks ✓ - Tree remains collapsed for clean view - Expand any node to see its checked children - Modify selections as needed and save again

### Tab 2: Manage Configurations

**Purpose**: Create and edit node extraction rules.

#### View Existing Configurations

The table shows all configured nodes:

| Section Path | Version | Message | Enabled | Priority | Has Expected Refs |
| --- | --- | --- | --- | --- | --- |
| PassengerList | 19.2 | OrderViewRS | ✓ | high | Yes |
| SegmentList | 19.2 | OrderViewRS | ✓ | high | Yes |

## Discovery Workflow

Discovery is the core workflow that extracts structure and relationships from XML.

### When to Run Discovery

**For Existing Airlines**: - Analyzing known/existing airline XML formats - Creating pattern library from existing airline data - Documenting standard XML structures

**Periodic Updates**: - XML structure changes in existing airlines - New fields added to existing patterns - Updating pattern library

### Step-by-Step: Running Discovery

#### Step 1: Access Discovery Page

Click **🔍 Discovery** in the sidebar

#### Step 2: Select Workspace

Workspace: [Dropdown: default, WestJet, United, ...]

* Select existing workspace or create new one
* Each workspace isolates data per airline/project
* See [Workspaces](#workspaces) section for details

#### Step 3: Upload XML File

**Option A: Drag & Drop** - Drag XML file into the upload area

**Option B: Browse** - Click **Browse** button - Select XML file from your computer

**Supported Formats**: - ✅ .xml files - ✅ Any messages (OrderViewRS, AirShoppingRS, etc.) - ✅ With or without IATA\_ prefix - ✅ Any version (17.2, 18.1, 19.2, 21.3)

**File Size Limits**: - Maximum: 100 MB - Recommended: < 10 MB for faster processing

#### Step 4: Review Upload Confirmation

✅ File uploaded successfully!  
  
File: WestJet\_OrderView\_19.2.xml  
Size: 2.3 MB  
Message Type: OrderViewRS  
NDC Version: 19.2  
Airline: WS (WestJet)

#### Step 5: Start Discovery

Click **🚀 Start Discovery** button

**What Happens Next**:

1. **Version Detection** (5 seconds)

* 🔍 Detecting NDC version and message type...

1. **Node Extraction** (1-3 minutes)

* 📥 Extracting node structures from XML...  
  Progress: [=========> ] 45%
  + Parses XML into subtrees
  + Sends subtrees to LLM for analysis
  + LLM extracts structure information

1. **Relationship Analysis** (1-2 minutes)

* 🔗 Analyzing relationships between nodes...
  + Discovers references between nodes
  + Validates expected references
  + Classifies relationships

1. **Pattern Generation** (30 seconds)

* 🎨 Generating reusable patterns...
  + Creates pattern templates from extracted node structures
  + Deduplicates similar patterns
  + Stores for future Identify runs

1. **Completion**

* ✅ Discovery completed successfully!

#### Step 6: Review Results

After Discovery completes, you’ll see comprehensive results:

### Discovery Results: Run Summary

📊 Discovery Results  
  
Run ID: 7322e7bb-fda6-4544-9bf3-cf2bc8e6e476  
Status: ✅ Completed  
Duration: 3m 42s  
  
NDC Version: 19.2  
Message Root: OrderViewRS  
Airline: WS (WestJet)

### Discovery Results: Statistics

📈 Extraction Statistics  
  
Nodes Analyzed: 47  
Relationships Found: 18  
Patterns Generated: 12  
  
Breakdown by Section:  
├─ PassengerList: 8 nodes  
├─ SegmentList: 6 nodes  
├─ FareList: 12 nodes  
├─ ServiceList: 5 nodes  
└─ Order: 16 nodes

### Discovery Results: Relationships

🔗 Relationship Summary  
  
✅ Valid Relationships: 15  
❌ Broken Relationships: 3  
📋 Expected Validated: 12  
⚠️ Expected Missing: 2  
🔍 Unexpected Discovered: 4

**Expand to see details**:

**Expected & Validated** ✅📋:

PassengerList → SegmentList  
Reference: SegmentRefID  
Status: Valid  
Confidence: 95%

**Expected but Missing** ❌📋:

PassengerList → ServiceList  
Reference: ServiceRefID (expected but not found)  
Status: Broken

**Unexpected Discoveries** ✅🔍:

PassengerList → BaggageList  
Reference: BaggageRefID  
Status: Valid (newly discovered)  
Confidence: 87%  
Note: This reference was not configured but AI found it

### Discovery Results: Generated Patterns

🎨 Generated Patterns: 12  
  
Patterns are now available in Pattern Manager for:  
- Future validation (Identify workflow)  
- Export/documentation  
- Cross-airline comparison

Click **View in Pattern Manager** to see patterns.

### Common Discovery Scenarios

#### Scenario 1: Creating Patterns from Existing Airline

**Situation**: Analyzing an existing airline’s XML to create reusable patterns.

**Steps**: 1. Create new workspace: Airline\_Code (e.g., WestJet) 2. Configure nodes in Node Config (select which nodes to extract) 3. Upload existing airline XML file 4. Run Discovery 5. Review generated patterns and relationships

**Result**: Pattern library created for this existing airline, ready for future Identify operations with new airlines.

#### Scenario 2: Updating Existing Patterns

**Situation**: Existing airline’s XML structure has changed, need to update patterns.

**Steps**: 1. Use the existing workspace for that airline (e.g., WestJet) 2. Upload updated XML file with new structure 3. Run Discovery 4. Compare: - New relationships discovered = structural additions - Missing expected references = structural changes

**Result**: Updated pattern library reflecting new structure.

#### Scenario 3: Validating New Airline (Use Identify Instead)

**Situation**: New airline XML needs validation against existing patterns.

**Action**: Use **Identify Workflow** instead of Discovery.

**Steps**: 1. Use workspace containing patterns from existing airlines 2. Upload new airline’s XML file 3. Run **Identify** (not Discovery) 4. Review pattern matches to see how closely new airline matches existing patterns

**Result**: Compatibility report showing how well new airline conforms to known patterns.

## Pattern Manager

After Discovery generates patterns, manage them in Pattern Manager.

### Access Pattern Manager

Click **🎨 Pattern Manager** in sidebar

### View Patterns

**Pattern Table**:

| Section Path | Node Type | Version | Airline | Message | Must-Have Attrs | Has Children |
| --- | --- | --- | --- | --- | --- | --- |
| PassengerList | Passenger | 19.2 | LA | OrderViewRS | 5 | ✓ |
| SegmentList | Segment | 19.2 | LA | OrderViewRS | 3 | ✓ |

**Summary Metrics**:

Total Patterns: 12  
Versions: 1 (19.2)  
Node Types: 8

### Filter Patterns

**Filter by Version**:

Version: [All ▼]  
Options: All, 17.2, 18.1, 19.2, 21.3

**Filter by Node Type**:

Node Type: [All ▼]  
Options: All, Passenger, Segment, Fare, Service...

### Export Patterns

**Purpose**: Share patterns with team, backup, or import to another workspace.

**Steps**:

1. **Select Patterns to Export**:
   * Check the **Select** checkbox for patterns you want
   * Or select all
2. Click **📤 Export Selected Patterns (JSON)**
3. **Downloaded File**:

* patterns\_LATAM\_2025-10-09.json

**Use Cases**: - Documentation: Include in API specs - Backup: Save pattern library - Sharing: Send to other team members - Migration: Import to different workspace

### Import Patterns

**Purpose**: Load patterns from another workspace or external source.

**Steps**:

1. Click **📥 Import Patterns**
2. **Upload JSON File**:
   * Must be valid pattern export file
   * Format: Same as export
3. **Preview**:

* Importing 12 patterns from LATAM workspace  
  Spec Version: 19.2  
  Message Root: OrderViewRS

1. Click **Confirm Import**
2. **Result**:

* ✅ Successfully imported 12 patterns

**Notes**: - Duplicate patterns (same signature\_hash) are skipped - Import doesn’t overwrite existing patterns

### Pattern Details

Click any pattern row to see full details:

**Pattern Information**:

ID: 42  
Section Path: OrderViewRS/Response/DataLists/PassengerList  
Node Type: Passenger  
Version: 19.2 / OrderViewRS  
Airline: LA

**Decision Rule**:

{  
 "node\_type": "Passenger",  
 "must\_have\_attributes": ["PaxID", "GivenName", "Surname"],  
 "optional\_attributes": ["MiddleName", "Title"],  
 "must\_have\_children": ["ContactInfo"],  
 "optional\_children": ["FrequentFlyer", "SSR"],  
 "child\_structure": {  
 "has\_children": true,  
 "min\_children": 1,  
 "max\_children": 20  
 }  
}

**Examples**:

Sample XML snippets that match this pattern

## Identify Workflow

Identify validates new XML files against existing patterns.

### When to Use Identify

**For New/Unknown Airlines**: - You’ve run Discovery on existing airlines and generated patterns - Now you want to validate new airline XML files against those patterns

**Use Cases**: - New Airline Validation: Check how closely a new airline’s XML matches existing patterns - Compliance Testing: Verify new airline follows standard structures - Deviation Detection: Identify differences from known patterns - Onboarding: Assess compatibility of new airline XML with existing systems

### Prerequisites

✅ **Patterns must exist** in workspace (from Discovery) ✅ **Patterns must match XML type** (same message\_root)

### Step-by-Step: Running Identify

#### Step 1: Access Identify Page

Click **🎯 Identify** in sidebar

#### Step 2: Select Workspace

Workspace: [Dropdown]

Same workspace where you ran Discovery.

#### Step 3: Upload XML File

* Upload the XML file you want to validate
* Must be same message type as patterns (e.g., OrderViewRS)

**Version Compatibility**: - ✅ Same version as patterns: Full validation - ⚠️ Different version: Partial validation (structure may differ)

#### Step 4: Start Identify

Click **🔍 Start Identify**

**What Happens**:

1. **Extract Node Structures** (1-2 minutes)
   * Same extraction as Discovery
   * Analyzes relationships between nodes
   * No pattern generation
2. **Match Against Patterns** (30 seconds)
   * Compare each extracted node to existing patterns
   * Compare relationships against expected patterns
   * Calculate similarity scores
   * Classify matches
3. **Generate Report** (instant)

### Identify Results

#### Pattern Matching Summary

📊 Pattern Matching Results  
  
Total Nodes Analyzed: 45  
Match Rate: 87.5%  
  
Verdict Breakdown:  
✅ EXACT\_MATCH: 32 (71%)  
🟡 HIGH\_MATCH: 7 (16%)  
🟠 PARTIAL\_MATCH: 3 (7%)  
⚪ NO\_MATCH: 2 (4%)  
🔴 NEW\_PATTERN: 1 (2%)

**What the verdicts mean**:

**EXACT\_MATCH** ✅ (Confidence ≥ 95%): - Node perfectly matches known pattern - All must-have attributes present - Structure identical

**HIGH\_MATCH** 🟡 (Confidence 85-95%): - Close match to pattern - Minor differences (optional fields) - Generally acceptable

**PARTIAL\_MATCH** 🟠 (Confidence 70-85%): - Some differences from pattern - Missing some expected fields or extra fields - Review recommended

**LOW\_MATCH/NO\_MATCH** ⚪ (Confidence < 70%): - Significant deviation from pattern - Structure changed - Investigation required

**NEW\_PATTERN** 🔴: - No matching pattern found - Completely new node structure - Consider running Discovery

#### Pattern Matches Table

| Node Type | Section Path | Explanation | Confidence | Verdict |
| --- | --- | --- | --- | --- |
| Passenger | PassengerList | Perfect match: All expected fields present | 100% | EXACT\_MATCH ✅ |
| Segment | SegmentList | Close match: Optional field ‘OperatingCarrier’ missing | 89% | HIGH\_MATCH 🟡 |
| Service | ServiceList | No matching pattern found | N/A | NEW\_PATTERN 🔴 |

#### Detailed Match Analysis

**Click any match to see detailed comparison**:

**Match Summary**:

Node: Passenger at /PassengerList  
Pattern ID: 42  
Verdict: HIGH\_MATCH 🟡  
Confidence: 89%

**Quick Explanation**:

✅ Strong match: 'Passenger' closely matches the expected pattern  
with 89% confidence. Optional field 'MiddleName' is present but not required.

**Detailed Comparison**:

**Attributes**:

✅ Matched: PaxID, GivenName, Surname, PaxRefID  
✅ Extra (OK): MiddleName (optional field present)  
❌ Missing: (none)

**Children**:

✅ Matched: ContactInfo  
✅ Extra: FrequentFlyer (bonus data)  
❌ Missing: (none)

**References**:

✅ Matched: SegmentRefID → SegmentList  
✅ Matched: FareRefID → FareList

**Full JSON Comparison**: - Expand to see side-by-side comparison of extracted node vs known pattern

#### Get AI Explanation

**For complex deviations**, click **🤖 Get Detailed AI Explanation**:

🤖 AI Analysis  
  
The Passenger node in this XML file is structurally similar to the  
known pattern, with 89% confidence. The main differences are:  
  
1. Additional Field: 'MiddleName' is present but was marked optional  
 in the pattern. This is acceptable and common for middle name variations.  
  
2. All required fields are present: PaxID, GivenName, Surname match  
 the pattern requirements exactly.  
  
3. References are valid: Both SegmentRefID and FareRefID correctly  
 point to existing nodes in the XML.  
  
Recommendation: This is a HIGH\_MATCH and acceptable variation. No  
action needed unless 'MiddleName' should be mandatory.

**Cached**: Explanations are cached for performance.

### Identify Workflow Tips

**Best Practices**:

1. **Run Discovery First**: Always have patterns before Identify
2. **Same Message Type**: Identify OrderViewRS against OrderViewRS patterns
3. **Review NEW\_PATTERN**: These may indicate structure changes
4. **Investigate NO\_MATCH**: Could be data quality issues

**Common Issues**:

**“No pattern matches found”**: - Cause: No patterns exist for this message type - Solution: Run Discovery first to generate patterns

**Low confidence scores**: - Cause: XML structure changed since Discovery - Solution: Review differences, possibly re-run Discovery

## Workspaces

Workspaces isolate data per airline or project. For basic workspace creation, see [Quick Start: Create a Workspace](#quick-start-create-a-workspace) in the Getting Started section. For detailed management, see [Managing Workspaces](#managing-workspaces) in the Configuration section.

### What is a Workspace?

A **workspace** is an isolated environment containing: - Patterns - Relationships - Node Configurations - Discovery/Identify runs

**Benefits**: - ✅ **Separation**: LATAM data separate from United data - ✅ **Organization**: One workspace per project - ✅ **Clean Comparison**: Compare airlines side-by-side

## Troubleshooting

### Understanding Error Messages

AssistedDiscovery provides detailed error messages to help you diagnose issues quickly. When an error occurs, you’ll see:

1. **Error Description**: What went wrong
2. **Error Details**: Specific technical information
3. **Troubleshooting Tips**: Context-specific suggestions
4. **Log File Location**: Where to find detailed logs

**Example Error Message**:

❌ Failed to analyze XML structure  
\*\*Error:\*\* argument of type '\_cython\_3\_1\_4.cython\_function\_or\_method' is not iterable  
Status Code: 500  
  
💡 Troubleshooting:  
- Check if the XML file is well-formed and valid  
- Verify it's a supported message type (e.g., AirShoppingRS)  
- Check the log files for detailed error information:  
  
📂 Log File (macOS):  
~/Library/Logs/AssistedDiscovery/assisted\_discovery.log

### How to Use Log Files

When you see an error, always check the log files for detailed information:

**Accessing Logs:**

**Option 1: Via UI** (Recommended) 1. Go to **⚙️ Config** page 2. Scroll to **📋 Application Logs** section 3. Click **📂 Open Log Folder** button 4. Open assisted\_discovery.log in a text editor

**Option 2: Direct File Access** - **macOS**: ~/Library/Logs/AssistedDiscovery/assisted\_discovery.log - **Windows**: %LOCALAPPDATA%\AssistedDiscovery\Logs\assisted\_discovery.log - **Linux**: ~/.local/share/AssistedDiscovery/logs/assisted\_discovery.log

**What to Look For in Logs:** - **Error timestamps**: Match with when your operation failed - **Run IDs**: Unique identifier for each Discovery/Identify run - **Stack traces**: Detailed error information - **LLM API responses**: Check for API-specific errors

**Tip**: When reporting issues, always include: - The error message shown in UI - Relevant log file excerpts (with timestamps) - The Run ID (if available) - What you were trying to do

### LLM Connection Issues

#### “Connection error” or “Cannot connect to endpoint”

**Causes**: - Invalid API key - Incorrect endpoint URL - Network/proxy issues - SSL certificate issues (corporate proxy)

**Solutions**:

1. **Verify Configuration**:
   * Go to ⚙️ Config → LLM Configuration
   * Click **🔍 Test Connection**
   * Check error message
2. **Check Endpoint Format**:

* ✅ Correct: https://your-resource.openai.azure.com/  
  ❌ Wrong: https://your-resource.openai.azure.com (missing trailing /)  
  ❌ Wrong: your-resource.openai.azure.com (missing https://)

1. **Verify API Key**:
   * Check Azure Portal for correct key
   * Key should be ~40+ characters
   * No spaces or line breaks
2. **Check Logs**:
   * Open log folder (Config → Application Logs)
   * Look for detailed error messages
   * Check for “SSL” or “certificate” errors
3. **Corporate Proxy**:
   * If behind corporate proxy, SSL verification is disabled (built-in)
   * If still failing, contact IT for proxy whitelist

#### “LLM INITIALIZATION FAILED: No API keys found”

**Cause**: .env file not loaded or missing credentials

**Solutions**:

1. **Configure via UI**:
   * Go to ⚙️ Config → LLM Configuration
   * Enter credentials
   * Save Configuration
   * **Restart backend** (important!)
2. **Check .env file** (advanced):

* # Check if file exists  
  ls -la .env  
    
  # View contents (backend directory)  
  cat .env | grep AZURE\_OPENAI\_KEY

1. **Restart Backend**:
   * Configuration changes require restart
   * Stop and restart backend service

### XML Analysis Issues

#### “Failed to analyze XML structure”

**When it happens**: Uploading XML to Node Configuration Manager

**What you’ll see**:

❌ Failed to analyze XML structure  
\*\*Error:\*\* [Technical error message from backend]  
  
💡 Troubleshooting:  
- Check if the XML file is well-formed and valid  
- Verify it's a supported message type (e.g., AirShoppingRS)  
- Check the log files for detailed error information:  
  
📂 Log File (macOS):  
~/Library/Logs/AssistedDiscovery/assisted\_discovery.log

**Solutions**:

1. **Validate XML Format**:
   * Open XML in text editor
   * Check for:
     + Unclosed tags
     + Special characters (&, <, >)
     + Encoding issues
     + Malformed structure
   * Use online XML validator: https://www.xmlvalidation.com/
2. **Check File Type**:
   * Must be NDC XML message (AirShoppingRS, OrderViewRS, etc.)
   * Not just any XML file
   * Should have NDC namespace declarations
3. **Review Log File**:
   * Go to **⚙️ Config** → **📋 Application Logs**
   * Click **📂 Open Log Folder**
   * Search for error timestamp in assisted\_discovery.log
   * Look for detailed Python stack trace
4. **Try Different XML**:
   * If one file fails, try another sample
   * Use a known-good XML file first
   * Check if issue is file-specific or systemic
5. **Report Issue**:
   * If problem persists, report to development team
   * Include:
     + Error message from UI
     + Log file excerpt (with timestamp)
     + Sample XML file (if not sensitive)

### Discovery/Identify Issues

#### “No patterns extracted”

**Causes**: - XML file empty or corrupted - LLM not configured - XML format not supported - No nodes configured for extraction

**Solutions**:

1. **Verify XML File**:
   * Open in text editor
   * Check it’s valid XML
   * Ensure it’s NDC format (not random XML)
2. **Check LLM**:
   * Test LLM connection in Config
   * Check logs for LLM errors
3. **Check Node Configurations**:
   * If configured, ensure nodes are “Enabled”
   * Try disabling configs to allow auto-detection

#### “No patterns found” when running Identify

**Cause**: No patterns exist for this message type/version

**Solution**: Run Discovery first to generate patterns

**Example**:

Patterns in workspace: 19.2/OrderViewRS  
Your XML: 21.3/AirShoppingRS  
  
❌ Mismatch: Different message types

**Fix**: Either: - Run Discovery on AirShoppingRS to create patterns - Use OrderViewRS file for Identify

#### “Relationship analysis failed”

**Causes**: - LLM timeout - Invalid XML snippets - Network interruption

**Solutions**:

1. **Check Logs**: Look for specific error
2. **Retry**: Re-run Discovery
3. **Smaller XML**: Try with smaller sample file first
4. **Increase Timeout**: Contact administrator if persistent

### UI Issues

#### “Backend connection failed”

**Causes**: - Backend not running - Wrong port - Firewall blocking

**Solutions**:

1. **Check Backend Status**:

* # Check if backend is running  
  curl http://localhost:8000/health  
    
  # Expected response:  
  {"status": "healthy"}

1. **Start Backend**:

* # Portable distribution  
  ./start\_app.sh  
    
  # Development  
  uvicorn app.main:app --host 127.0.0.1 --port 8000 --reload

1. **Check Port**:
   * Backend should be on port 8000
   * Frontend should be on port 8501
   * Ensure no conflicts

#### Page not loading or spinning forever

**Causes**: - Backend slow response - Large data set - Browser cache

**Solutions**:

1. **Wait**: Discovery/Identify can take 3-5 minutes
2. **Check Logs**: Backend logs show progress
3. **Refresh**: Ctrl+F5 (hard refresh)
4. **Clear Cache**: Browser → Clear cache and cookies

### Data Issues

#### “Workspace not found” after switching

**Cause**: Workspace database not created yet

**Solution**: Create workspace first in Config page

#### “Patterns still show after deletion”

**Cause**: Frontend cache

**Solution**: Refresh page (F5 or Ctrl+R)

### Performance Issues

#### Discovery is very slow (>10 minutes)

**Causes**: - Large XML file (>10MB) - Many node types (>50) - Slow LLM API responses

**Solutions**:

1. **Configure Nodes**: Only extract important nodes
2. **Smaller Sample**: Use smaller XML file for initial Discovery
3. **Check Logs**: See which step is slow

#### UI becomes unresponsive

**Causes**: - Large result set (>1000 rows) - Browser memory - Slow rendering

**Solutions**:

1. **Use Pagination**: Limit results to 50-100 per page
2. **Filter Data**: Apply filters before viewing
3. **Restart Browser**: Clear memory

## Best Practices

### Discovery Workflow

**Do**: - ✅ Configure expected references before Discovery - ✅ Use representative XML samples (not minimal test files) - ✅ Review unexpected discoveries (they may reveal issues) - ✅ Run Discovery when XML format changes - ✅ Use one workspace per airline/project

**Don’t**: - ❌ Skip node configuration for important nodes - ❌ Run Discovery on every single file (use Identify instead) - ❌ Ignore broken references (investigate root cause) - ❌ Mix airlines in same workspace

### Pattern Management

**Do**: - ✅ Export patterns for backup - ✅ Document pattern changes - ✅ Review patterns periodically - ✅ Share patterns with team

**Don’t**: - ❌ Delete patterns without backup - ❌ Manually edit pattern JSON (use Discovery) - ❌ Import untrusted patterns

### Identify Workflow

**Do**: - ✅ Run Identify on test files before production - ✅ Investigate NEW\_PATTERN verdicts - ✅ Use Identify for regression testing - ✅ Archive identify results for audit trail

**Don’t**: - ❌ Ignore low confidence matches - ❌ Run Identify without patterns - ❌ Skip AI explanations for deviations

### Workspace Management

**Do**: - ✅ Create workspace per airline - ✅ Use clear naming conventions - ✅ Switch workspace before operations - ✅ Backup workspace data (export patterns)

**Don’t**: - ❌ Delete workspace without backup - ❌ Mix unrelated data in one workspace - ❌ Forget to switch workspace

### Configuration

**Do**: - ✅ Test LLM connection after configuration - ✅ Restart backend after config changes - ✅ Document configuration settings - ✅ Set expected references for known node types

**Don’t**: - ❌ Share API keys - ❌ Use production keys in test environment - ❌ Skip configuration validation

## Appendix

### Supported NDC Versions

* ✅ NDC 17.2
* ✅ NDC 18.1
* ✅ NDC 19.2
* ✅ NDC 21.3
* ✅ Any future version (auto-detected)

### Supported Message Types

* ✅ OrderViewRS / IATA\_OrderViewRS
* ✅ AirShoppingRS / IATA\_AirShoppingRS
* ✅ OfferPriceRS / IATA\_OfferPriceRS
* ✅ OrderCreateRQ / IATA\_OrderCreateRQ
* ✅ OrderChangeRQ / IATA\_OrderChangeRQ
* ✅ Any NDC message type (dynamic support)

### Supported Formats

* ✅ With IATA\_ prefix (NDC 19.2+)
* ✅ Without IATA\_ prefix (NDC 17.2)
* ✅ Mixed formats in same XML

### System Requirements

**Minimum**: - 4 GB RAM - 2 CPU cores - 1 GB disk space - Python 3.9+

**Recommended**: - 8 GB RAM - 4 CPU cores - 5 GB disk space - Python 3.10+

### Browser Compatibility

* ✅ Chrome/Edge (Recommended)
* ✅ Firefox
* ✅ Safari
* ❌ Internet Explorer (not supported)

## Getting Help

### Documentation

* **This User Guide**: Complete reference
* **Relationship Discovery Logic**: Technical deep-dive
* **Packaging Guide**: Deployment instructions
* **API Documentation**: Backend API reference

### Support Channels

**Issues & Bugs**: - GitHub Issues: <repository-url>/issues

**Questions**: - Team Chat: Contact development team - Email: <support-email>

### Log Files

Always include log files when reporting issues:

**Location**: - macOS: ~/Library/Logs/AssistedDiscovery/ - Windows: %LOCALAPPDATA%\AssistedDiscovery\Logs\ - Linux: ~/.local/share/AssistedDiscovery/logs/

**Access via UI**: 1. Go to ⚙️ Config 2. Scroll to 📋 Application Logs 3. Click 📂 Open Log Folder

## Frequently Asked Questions

**Q: Can I use AssistedDiscovery offline?** A: No, it requires internet connection for LLM API calls (Azure OpenAI or Gemini).

**Q: Can multiple users share workspaces?** A: Not directly. Export/import patterns to share data between users. In the future, a centralised database will be used to share data between users.

**Q: Can I modify extracted patterns?** A: Run Discovery again with refined configuration.

**Q: How accurate is relationship discovery?** A: LLM-based discovery has ~85-95% accuracy. Always validate unexpected discoveries and review confidence scores.

**Q: Why do I get different results when running Discovery twice on the same XML?** A: AssistedDiscovery uses AI (LLMs) which are non-deterministic. Each run may produce slightly different results due to: - Random sampling in AI models - Different interpretation of ambiguous structures - Temperature settings (we use low temperature for consistency, but not zero)

**Recommendation**: If results vary significantly: - Check confidence scores - trust high confidence (>90%) more - Run multiple times and look for consistent patterns - Review differences manually - Report significant inconsistencies to help improve the system

**Q: Can LLMs make mistakes?** A: Yes, absolutely. LLMs can: - Miss relationships that exist - Find relationships that don’t exist - Misinterpret node structures - Generate incorrect patterns

**Always review results**, especially: - Low confidence matches (< 85%) - Unexpected discoveries - Broken relationships - New patterns

Think of AssistedDiscovery as an **intelligent assistant**, not a perfect oracle. Human validation is essential.

**Q: How can I improve accuracy?** A: Several ways: 1. **Configure expected references** in Node Configuration before Discovery 2. **Use representative XML samples** (not edge cases) 3. **Review and validate** all unexpected discoveries 4. **Run multiple times** and compare for consistency 5. **Provide feedback** when AI makes mistakes 6. **Use clear, well-formed XML** files

**Q: What should I do if I find an error in the results?** A: Please report it! Your feedback helps improve the system: 1. Note the Run ID from the results page 2. Save the error details and confidence scores 3. Export relevant patterns/results 4. Include log files if possible