

LanguageDataExporter

User Guide

Version 1.0.16

Language XML to Categorized Excel Converter
with VRS-based Story Ordering

Table of Contents

- [1. Quick Start](#)
- [2. Installation](#)
- [3. GUI Mode](#)
- [4. CLI Mode](#)
- [5. Category System](#)
- [6. VRS Ordering](#)
- [7. Word Count Reports](#)
- [8. Output Files](#)
- [9. Troubleshooting](#)

1. Quick Start

30-Second Workflow

1. Double-click `LanguageDataExporter.exe`
2. Click `Generate Language Excels` to create files

3. Find output in **GeneratedExcel/** folder

Optional: Click "Analyze Categories" first to preview the category distribution before exporting.

What It Does



Feature	Description	Output
Language Export	Convert XML to categorized Excel	LanguageData_{LANG}.xlsx
Word Count Report	LQA scheduling metrics	WordCountReport.xlsx
VRS Ordering	Chronological story order	Sorted STORY rows
Two-Tier Clustering	STORY + GAME_DATA categories	Color-coded cells

2. Installation

Requirements

Requirement	Details
OS	Windows 10/11
Disk Space	~50 MB
Network	Access to game data folders
Drive	Perforce sync on D:, E:, or F:

Installation Steps

- 1 **Download** - Get [LanguageDataExporter-Setup.exe](#) from GitHub Releases
- 2 **Run Installer** - Double-click and follow the prompts
- 3 **Configure Drive Letter**

Drive Configuration

On first launch, select your Perforce drive letter:

F: (Default) - Most common
D: or **E:** - Alternative drives

This sets paths to LOC, EXPORT, and VRS folders.

- 4 **Launch** - Double-click [LanguageDataExporter.exe](#)

Folder Structure

```
LanguageDataExporter/
├── LanguageDataExporter.exe ← Main application
├── settings.json ← Your drive configuration
├── category_clusters.json ← Category colors/keywords
├── GeneratedExcel/ ← Output folder
│   ├── LanguageData_ENG.xlsx
│   ├── LanguageData_FRE.xlsx
│   ├── WordCountReport.xlsx
│   └── _Summary.xlsx
└── _internal/ ← Python runtime
```

3. GUI Mode

Launch by double-clicking **LanguageDataExporter.exe**

Interface Layout

LanguageDataExporter

CONFIGURED PATHS

```
| LOC Folder: F:\perforce\...\loc [OK] |
| EXPORT Folder: F:\perforce\...\export [OK] |
| Output Folder: GeneratedExcel [OK] |
```

CATEGORY ANALYSIS

[Analyze Categories]		
Category	Files	Tier
Sequencer	340	STORY
Item	340	GAME_DATA

EXPORT ACTIONS

[Generate Word Count Report]

[Generate Language Excels]

GUI Actions

Button	What It Does	Output
Analyze Categories <small>(Optional)</small>	Informational only - Scans EXPORT folder and shows category distribution. Not required for export (export does this automatically).	TreeView updated

**Generate
Word Count
Report**

Creates LQA metrics report

WordCountReport.xlsx

**Generate
Language
Excels**

Creates all language files

LanguageData_*.xlsx

4. CLI Mode

Basic Commands

```
# Run with GUI (default)
python main.py

# Run in CLI mode
python main.py --cli

# Process specific languages
python main.py --cli --lang eng,fre,ger

# Generate word count report
python main.py --cli --word-count

# Preview without writing files
python main.py --cli --dry-run

# Show category distribution
python main.py --list-categories
```

CLI Arguments Reference

Argument	Description	Example
--cli	Run in command-line mode	--cli
--lang	Process specific languages	--lang eng,fre
--word-count	Include word count report	--word-count
--word-count-only	Only generate word count report	--word-count-only
--dry-run	Preview without writing	--dry-run
--list-categories	Show category distribution	--list-categories

--output	Custom output folder	--output D:\Out
-v	Enable debug logging	-v

5. Category System (THE ALGORITHM)

This section explains the **complete category clustering algorithm** - the core logic that determines which category each string belongs to.

This is the Most Important Section!

Understanding the algorithm helps you predict exactly which category any file will be assigned to.

Algorithm Overview

CATEGORY CLUSTERING ALGORITHM

INPUT: File path from EXPORT folder

STEP 1: DETERMINE TIER

Is file in Dialog/ or Sequencer/ folder?

YES → **TIER 1 (STORY)** → Folder-based categorization

NO → **TIER 2 (GAME_DATA)** → Two-phase keyword matching

STEP 2A: TIER 1 - STORY

Sequencer/ → Sequencer

Dialog/AIDialog/ → AIDialog

Dialog/QuestDialog/ → QuestDialog

Dialog/NarrationDialog/ → NarrationDialog

STEP 2B: TIER 2 - GAME_DATA (Two-Phase)

PHASE 1: PRIORITY KEYWORDS (checked FIRST!)

gimmick → Gimmick | item → Item | quest → Quest

skill → Skill | character → Character

faction → Faction | region → Region

IF MATCH FOUND → RETURN IMMEDIATELY

PHASE 2: FOLDER + KEYWORD PATTERNS

(Only if Phase 1 didn't match)

OUTPUT: Category name

Step 1: Tier Classification

Top-Level Folder	Tier	Processing Method
Dialog/	TIER 1 (STORY)	Subfolder determines category
Sequencer/	TIER 1 (STORY)	All files → Sequencer
System/	TIER 2 (GAME_DATA)	Two-phase keyword matching
World/	TIER 2 (GAME_DATA)	Two-phase keyword matching
None/ , Platform/	TIER 2 (GAME_DATA)	Two-phase keyword matching

Step 2A: TIER 1 - STORY Categories

STORY content uses simple folder-based categorization and is sorted chronologically using VRS.

Folder Path	Category	Color	Ordering
Sequencer/*.loc.xml	Sequencer	Light Orange	VRS chronological
Dialog/AIDialog/*.loc.xml	AIDialog	Light Green	VRS chronological
Dialog/QuestDialog/*.loc.xml	QuestDialog	Light Green	VRS chronological
Dialog/NarrationDialog/*.loc.xml	NarrationDialog	Light Green	VRS chronological
Dialog/StageCloseDialog/*.loc.xml	QuestDialog (mapped)	Light Green	VRS chronological

Step 2B: TIER 2 - GAME_DATA Two-Phase Matching

This is the core algorithm for non-story content. It uses two phases, checked in order.

PHASE 1: Priority Keywords (CHECKED FIRST!)

CRITICAL: Priority keywords completely override folder location!

A file named `KnowledgeInfo_Item.xml` in the `Knowledge/` folder will be categorized as **Item**, not Knowledge, because "item" is found in the filename.

The algorithm extracts the filename and checks if it contains any priority keyword. **First match wins and immediately returns.**

Priority	Keyword	Category	Example Match
1	gimmick	Gimmick	<code>gimmickinfo_item_book</code> → Gimmick
2	item	Item	<code>KnowledgeInfo_Item</code> → Item
3	quest	Quest	<code>characterinfo_quest</code> → Quest
4	skill	Skill	<code>factioninfo_skill</code> → Skill
5	character	Character	<code>gimmickinfo_character</code> → Character
6	faction	Faction	<code>regioninfo_faction</code> → Faction
7	region	Region	<code>uiinfo_region</code> → Region

Matching is SUBSTRING-based and CASE-INSENSITIVE

"item" matches: `Item`, `item`, `KnowledgeInfo_Item`, `itemequip`

PHASE 2: Standard Patterns (Only if Phase 1 didn't match)

Match Type	Pattern	Category	Color

Folder	<code>lookat/</code> , <code>patterndescription/</code>	 Item	Light Purple
Keyword	<code>weapon</code> , <code>armor</code>	 Item	Light Purple
Folder	<code>quest/</code>	 Quest	Light Purple
Keyword	<code>schedule_</code>	 Quest	Light Purple
Folder	<code>character/</code> , <code>npc/</code>	 Character	Light Peach
Keyword	<code>monster</code> , <code>animal</code>	 Character	Light Peach
Folder	<code>skill/</code>	 Skill	Light Purple
Folder	<code>knowledge/</code>	 Knowledge	Light Purple
Folder	<code>faction/</code>	 Faction	Light Purple
Folder	<code>ui/</code>	 UI	Light Teal
Keyword	<code>localstringinfo</code> , <code>symboltext</code>	 UI	Light Teal
Folder	<code>region/</code>	 Region	Light Peach
(default)	(no match)	 System_Misc	Light Grey

Algorithm Walkthrough Examples

Example 1: File with "Item" keyword in Knowledge folder

World/Knowledge/KnowledgeInfo_Item.xml

Step 1: Top folder is "World/" → TIER 2 (GAME_DATA)

Step 2B Phase 1: Check "knowledgeinfo_item"

- "gimmick"? NO
- "item"? **YES** → RETURN "Item"

Result: Item (NOT Knowledge!)

Example 2: Gimmick file with multiple keywords**System/Gimmick/gimmickinfo_item_book.xml**

Step 1: Top folder is "System/" → TIER 2 (GAME_DATA)

Step 2B Phase 1: Check "gimmickinfo_item_book"

- "gimmick"? **YES** → RETURN "Gimmick"
- (Note: "item" is also present but gimmick is checked FIRST)

Result: Gimmick (gimmick has HIGHEST priority)

Example 3: File with no priority keywords**World/Knowledge/KnowledgeBase.xml**

Step 1: Top folder is "World/" → TIER 2 (GAME_DATA)

Step 2B Phase 1: Check "knowledgebase"

- gimmick? NO | item? NO | quest? NO | skill? NO
- character? NO | faction? NO | region? NO
- No match → Continue to Phase 2

Step 2B Phase 2: Check folder patterns

- "knowledge/" in path? **YES** → RETURN "Knowledge"

Result: Knowledge

Priority Keyword Conflict Resolution

When a filename contains **multiple** priority keywords, the **first match in priority order wins**:

Filename	Contains	Winner	Why
gimmickinfo_item_book	gimmick, item	Gimmick	gimmick is priority 1
characterinfo_quest	character, quest	Quest	quest is priority 3, character is 5
skillinfo_faction	skill, faction	Skill	skill is priority 4, faction is 6

Golden Rules Summary

Rule	Explanation
Tier First	Dialog/Sequencer → STORY, everything else → GAME_DATA
Priority Keywords Win	Phase 1 keywords override ALL folder matching
Gimmick is #1	"gimmick" in filename → always Gimmick category
Order Matters	Priority keywords checked in specific order (1-7)
Substring Match	Keywords match anywhere in filename (case-insensitive)
Knowledge is Catch-All	Only matches if NO priority keyword found

6. VRS Ordering

What is VRS?

VoiceRecordingSheet (VRS) is the master Excel file containing all voiced lines in chronological story order. LanguageDataExporter uses VRS to sort STORY content so LQA reviewers see dialogue in the order players experience it.

How It Works



Step	Action	Result
1	Load VoiceRecordingSheet.xlsx	Read EventName from Column W
2	Scan EXPORT XMLs	Extract SoundEventName attribute
3	Match StringID to EventName	Build ordering map
4	Sort STORY entries	Chronological story order!

Result: STORY strings appear in Excel in **chronological story order**. LQA reviewers see content as players experience it.

7. Word Count Reports

Report Purpose

Use Case	How It Helps
Schedule Work	Estimate LQA time based on word counts
Track Progress	Compare counts across languages
Find Untranslated	Identify strings still containing Korean

Counting Method

Language Type	Method	Languages
European/SEA	Word count	ENG, FRE, GER, SPA, POR, ITA, RUS, TUR, POL, THA, VIE, IND, MSA
CJK	Character count	JPN, ZHO-CN, ZHO-TW

Untranslated Detection

A string is marked **untranslated** if the translation still contains Korean characters (Unicode U+AC00-U+D7A3).

Example: "Hello 안녕" → Marked as untranslated

8. Output Files

Language Excel Files

Filename: **LanguageData_{LANG}.xlsx**

Column	Width	Description
StrOrigin	45	Korean source text
Str	45	Translated text
StringID	15	Unique identifier
English	45	English reference (EU languages only)
Category	20	Color-coded category

Note: CJK languages (JPN, ZHO-CN, ZHO-TW) don't include the English column.

9. Troubleshooting

Issue	Cause	Solution
Path NOT FOUND	Wrong drive letter	Edit <code>settings.json</code> or run <code>drive_replacer.py</code>
No language files	LOC folder empty	Check Perforce sync status
VRS not loaded	Missing VRS folder	Verify VRS path in settings
Empty output	No .loc.xml files	Check EXPORT folder exists
Wrong category	Priority keyword conflict	Check filename for keywords

Debug Mode

Run with `-v` flag for detailed logging:

```
python main.py --cli -v
```

LanguageDataExporter • Version 1.0.16 • January 2025

[GitHub Repository](#)