WORKFLOW_GUIDE

BIBLOS LOGOU Complete Workflow Guide

End-to-End Processing Workflow

This guide covers the complete workflow from raw data to finished commentary.

Table of Contents

---

1. System Initialization

1.1 Prerequisites Setup

```
# 1. Create Python virtual environment
python -m venv venv
source venv/bin/activate # Linux/macOS
# venv\Scripts\activate # Windows
# 2. Install dependencies
pip install -r requirements.txt
# 3. Configure environment
cp .env.example .env
# Edit .env with your PostgreSQL credentials
```

1.2 Database Initialization

```
# Create database
createdb -U postgres biblos_logou
# Initialize schema and all data
python main.py init --schema bible_refinement_db.sql --all
```

This creates:

* 16 core tables
* 73 canonical books
* 10 primary orbital motifs

* Hermeneutical principles
* Views, functions, and triggers

1.3 Verify Installation

```
python main.py status
```

Expected output:

```
================================================
BIBLOS LOGOU System Status
================================================
Table Counts:
canonical_books: 73
verses: 0
events: 0
motifs: 10
...
```

---

2. Data Ingestion

2.1 Verse Text Ingestion

From File

```
# Standard format: "Reference - Text"
python main.py ingest --verses data/kjv_verses.txt
```

From Bible API

```
# Populate verses with API text
python main.py fetch --populate --limit 1000
# Fetch specific book
python main.py fetch --populate --book "Genesis" --limit 500
```

2.2 Event Ingestion (Programmatic)

```
from scripts.ingestion import EventIngester
from scripts.database import init_db, get_db
init_db()
ingester = EventIngester(get_db())
events = [
{
'part_number': 1,
'part_title': 'BEFORE ALL THINGS',
'event_number': 1,
'event_description': 'Creation: Light from darkness'
},
{
'part_number': 1,
'part_title': 'BEFORE ALL THINGS',
'event_number': 2,
'event_description': 'Creation: Separation of waters'
},
# Continue for all events...
]
ingester.ingest_events(events)
```

2.3 Initialize Typological Network

```
# Initialize core Orthodox typological pairs
python main.py crossref --init-typology
```

---

3. Verse Processing Pipeline

3.1 Processing Stages

Each verse passes through:

```
RAW -> PARSED -> ANALYZED -> STRATIFIED -> FLESHED_OUT -> TONALLY_ADJUSTED -> REFINED -> VERIFIED
```

3.2 Basic Processing

```
# Process batch of 100 verses
python main.py process --batch 100

# Process specific verse
python main.py process --verse-id 1234

# Continuous processing until complete
python main.py process --continuous
```

3.3 Advanced Batch Orchestration

```
# View processing plan
python main.py orchestrate --plan sequential
python main.py orchestrate --plan by_category
python main.py orchestrate --plan incomplete_first

# Execute plan with parallel workers
python main.py orchestrate --execute by_category --workers 4 --batch-size 200

# Check/resume from checkpoints
python main.py orchestrate --list-checkpoints
python main.py orchestrate --run # Auto-resumes from checkpoint
```

3.4 Processing Components

Fourfold Sense Generation

```
from scripts.processing import FourfoldSenseGenerator

generator = FourfoldSenseGenerator()
senses = generator.generate_all_senses(verse, book_info)
# Returns: {literal, allegorical, tropological, anagogical}
```

Nine-Matrix Calculation

```
from scripts.processing import NineMatrixCalculator

calculator = NineMatrixCalculator()
matrix = calculator.calculate_all(verse, book_info)
# Returns: emotional_valence, theological_weight, narrative_function, etc.
```

Tonal Adjustment

```
from scripts.processing import TonalAdjuster

adjuster = TonalAdjuster()
tonal = adjuster.apply_tonal_adjustments(verse, book_info)
# Returns: tonal_weight, dread_amplification, local_emotional_honesty
```

---

4. Quality Assurance

4.1 Full Validation Suite

```
python main.py validate --full --sample-size 100
```

Checks:

* **Invisibility**: Ensures patterns remain subliminal
* **Thread Density**: Validates 18-22 bound compliance
* **Fourfold Sense**: Validates content quality
* **Content Quality**: Vocabulary diversity, sense balance

4.2 Specific Validations

```
# Validate specific verse
python main.py validate --verse-id 1234

# Check thread density at page
python main.py validate --density-page 500
```

4.3 Programmatic Validation

```
from scripts.validation import ValidationOrchestrator

validator = ValidationOrchestrator()

# Full validation
results = validator.run_full_validation()
print(f"Status: {results['overall']['status']}")

# Invisibility check
invis = validator.invisibility.verify_verse(verse_id)
print(f"Passes: {invis['passes']}, Score: {invis['overall_score']}")

# Thread density
density = validator.density.validate_page_range(0, 2500)
print(f"Pass rate: {density['pass_rate']}")
```

---

5. Output Generation

5.1 Progress Dashboard

```
python main.py export --dashboard
# Creates: output/Progress_Dashboard.md
```

5.2 Book Commentary Export

```
# Single book - Markdown
python main.py export --book "Genesis" --format markdown

# Single book - Both formats
python main.py export --book "Genesis" --format both

# All outputs
python main.py export --all --format both
```

5.3 Analytics Reports

```
# Generate analytics report
python main.py analytics --report --format both

# View processing velocity
```

```
python main.py analytics --processing
# View motif analytics
python main.py analytics --motifs
```

5.4 Output Files Generated

```
output/
+-- Progress_Dashboard.md
+-- Analytics_Report.md
+-- analytics_report.json
+-- Hermeneutical_Arrangement.md
+-- Motif_Registry.md
+-- Genesis_Commentary.md
+-- Genesis_Commentary.json
+-- full_database_export.json
```

---

6. Advanced Operations

6.1 Patristic Integration

```
# List Church Fathers
python main.py patristic --list-fathers
# Get Father information
python main.py patristic --father "John Chrysostom"
# Get commentary for verse
python main.py patristic --verse "Genesis 1:1"
# Generate catena
python main.py patristic --catena "John 1:1"
```

6.2 Cross-Reference Network

```
# Analyze verse references
python main.py crossref --analyze "Genesis 22:2"
# Get suggestions for sparse verses
python main.py crossref --suggest "Exodus 12:13"
# View network statistics
python main.py crossref --stats
```

6.3 Sensory Vocabulary Management

```
from tools.sensory_vocabulary import SensoryVocabularyManager
manager = SensoryVocabularyManager()
# Get vocabulary for verse
vocab = manager.get_vocabulary_for_verse(
category='gospel',
modality='visual',
current_page=1500,
count=3
)
# Get stats
stats = manager.get_vocabulary_stats()
```

6.4 AI-Enhanced Processing

```
from tools.ai_integration import AIEnhancedProcessor
processor = AIEnhancedProcessor()
# Generate senses with AI
```

```
    senses = processor.generate_fourfold_senses(
    verse_ref='John 1:1',
    verse_text='In the beginning was the Word...',
    book_category='gospel'
    )
    # Generate refined explication
    explication = processor.generate_refined_explication(
    verse_ref, verse_text, senses, matrix
    )
```

---

7. Maintenance & Monitoring

7.1 Regular Status Checks

```
# System status
python main.py status
# Analytics overview
python main.py analytics --processing
python main.py analytics --motifs
```

7.2 Validation Schedule

Run weekly:

```
python main.py validate --full --sample-size 200
```

7.3 Checkpoint Management

```
# List all checkpoints
python main.py orchestrate --list-checkpoints
# Resume processing from checkpoint
python main.py orchestrate --run
```

7.4 Database Maintenance

```
-- Check processing status distribution
SELECT status, COUNT(*) FROM verses GROUP BY status;
-- Find stalled processing
SELECT verse_reference, status, updated_at
FROM verses
WHERE status = 'failed'
ORDER BY updated_at DESC;
-- Check motif health
SELECT name, current_status, last_activation_page
FROM motifs
WHERE current_status != 'completed';
```

7.5 Log Monitoring

Logs are stored in `logs/`:

```
tail -f logs/biblos_logou_*.log
```

Quick Reference Card

Task	Command
Initialize system	`python main.py init --all`
Check status	`python main.py status`
Ingest verses	`python main.py ingest --verses file.txt`
Process batch	`python main.py process --batch 100`
Run validation	`python main.py validate --full`
Export dashboard	`python main.py export --dashboard`
Export book	`python main.py export --book "Genesis"`
View analytics	`python main.py analytics --report`
List Fathers	`python main.py patristic --list-fathers`
Check references	`python main.py crossref --analyze "Gen 1:1"`

Troubleshooting

Database Connection Issues

```
# Check PostgreSQL is running
pg_isready -h localhost -p 5432
# Verify credentials in .env
cat .env | grep BIBLOS_DB
```

Processing Stalls

```
# Check checkpoints
python main.py orchestrate --list-checkpoints
# Resume from checkpoint
python main.py orchestrate --run
```

Validation Failures

```
# Run detailed validation
python main.py validate --full --sample-size 50
python main.py -v validate --verse-id <failing_id>
```

Document Version: 2.0 *For BIBLOS LOGOU v2.0.0*