Universal Prompt & Reference System - Comprehensive Usage Examples

Real-world scenarios showcasing Port 42's context-aware tool and artifact creation

Overview

Port 42's Universal Prompt & Reference System allows you to create sophisticated tools and artifacts by combining: - **Custom AI prompts** (`--prompt`) - Specific instructions for AI generation - **Contextual references** (`--ref`) - Knowledge from files, tools, web content, and memories - **Transform capabilities** (`--transforms`) - What the tool should be able to do

Table of Contents

Development Tools

Code Analyzer with Custom Rules

```
# Create a code analyzer that understands your project structure
port42 declare tool project-analyzer --transforms analyze,code,lint \
  --ref file:./eslint.config.js \
  --ref file:./package.json \
  --ref file:./README.md \
  --prompt "Create a comprehensive code analyzer that checks TypeScript/JavaScript code agains

# Usage
project-analyzer ./src/
```

API Client Generator

```
# Generate API client from OpenAPI spec
port42 declare tool api-client-gen --transforms generate,api,client \
  --ref url:https://api.example.com/openapi.json \
  --ref file:./auth-config.json \
  --prompt "Generate a complete TypeScript API client with authentication, error handling, ret

# Usage
api-client-gen --output ./src/api/
```

Database Migration Helper

```
# Create migration tools with schema understanding
port42 declare tool db-migrator --transforms database,migrate,validate \
  --ref file:./schema.sql \
  --ref p42:/commands/db-connector \
  --ref search:"migration patterns" \
  --prompt "Build a smart database migration tool that validates schema changes, generates roll

# Usage
db-migrator generate --name add_user_preferences
db-migrator apply --dry-run
```

Data Processing

Log Analysis Pipeline

```
# Intelligent log processor with pattern recognition
port42 declare tool log-intelligence --transforms analyze,logs,patterns \
  --ref file:./nginx.conf \
  --ref file:./log-samples.txt \
  --ref url:https://nginx.org/en/docs/http/ngx_http_log_module.html \
  --prompt "Create an intelligent log analyzer that parses Nginx logs, detects anomalies, iden

# Create accompanying visualization tool
port42 declare tool log-visualizer --transforms visualize,dashboard \
  --ref p42:/commands/log-intelligence \
  --prompt "Build a real-time dashboard that displays log analysis results with charts, threat

# Usage
log-intelligence /var/log/nginx/access.log | log-visualizer --realtime
```

CSV Data Transformer

```
# Smart CSV processor with schema detection
port42 declare tool csv-transformer --transforms transform,csv,validate \
  --ref file:./data-schema.json \
  --ref file:./sample-data.csv \
  --ref search:"data validation patterns" \
  --prompt "Create a powerful CSV transformation tool that auto-detects schemas, validates data

# Usage
csv-transformer input.csv --schema auto --output clean-data.csv
```

DevOps & Infrastructure

Kubernetes Deployment Generator

```
# Context-aware K8s manifest generator
port42 declare artifact k8s-deployment --artifact-type config --file-type .yaml \
  --ref file:./app-config.json \
  --ref file:./Dockerfile \
  --ref url:https://kubernetes.io/docs/concepts/workloads/deployments/ \
  --prompt "Generate production-ready Kubernetes deployment manifests with auto-scaling, health

# Create deployment manager tool
port42 declare tool k8s-manager --transforms deploy,kubernetes,manage \
  --ref p42:/artifacts/k8s-deployment \
  --ref p42:/commands/kubectl-wrapper \
  --prompt "Build a deployment manager that validates manifests, handles rolling updates, moni

# Usage
k8s-manager deploy --environment production
k8s-manager status --watch
```

Infrastructure as Code Assistant

```
# Terraform module generator with best practices
port42 declare tool terraform-gen --transforms generate,terraform,infrastructure \
  --ref file:./terraform-modules/ \
  --ref file:./aws-requirements.md \
  --ref url:https://registry.terraform.io/providers/hashicorp/aws/latest/docs \
  --prompt "Create Terraform modules following AWS best practices with proper tagging, security

# Usage
terraform-gen --resource ec2 --environment prod --compliance sox
```

Documentation & Content

API Documentation Generator

```
# Comprehensive API docs with examples
port42 declare artifact api-documentation --artifact-type documentation \
  --ref p42:/commands/api-server \
  --ref file:./openapi.yaml \
  --ref url:https://spec.openapis.org/oas/v3.1.0 \
  --ref search:"API documentation best practices" \
  --prompt "Generate comprehensive API documentation with interactive examples, authentication

# Create interactive docs server
```

```
port42 declare tool docs-server --transforms serve,documentation,interactive \
  --ref p42:/artifacts/api-documentation \
  --prompt "Build a live documentation server with API testing capabilities, search functionali

# Usage
docs-server --port 8080 --watch
```

Code Documentation Extractor

```
# Intelligent code documentation
port42 declare tool doc-extractor --transforms extract,document,analyze \
  --ref file:./src/ \
  --ref file:./tsconfig.json \
  --ref p42:/commands/typescript-parser \
  --prompt "Extract and generate comprehensive documentation from TypeScript code including fun

# Usage
doc-extractor ./src --format markdown --output ./docs/api
```

Security & Monitoring

Security Scanner with Custom Rules

```
# Project-specific security scanner
port42 declare tool security-scanner --transforms scan,security,audit \
  --ref file:./package.json \
  --ref file:./security-policy.md \
  --ref url:https://owasp.org/www-project-top-ten/ \
  --ref search:"JavaScript security vulnerabilities" \
  --prompt "Create a comprehensive security scanner that checks for dependency vulnerabilities

# Create security reporter
port42 declare tool security-reporter --transforms report,security,dashboard \
  --ref p42:/commands/security-scanner \
  --prompt "Build a security reporting dashboard with risk scoring, remediation guidance, and

# Usage
security-scanner ./src --report json | security-reporter --format html
```

Performance Monitor

```
# Application performance analyzer
port42 declare tool perf-monitor --transforms monitor,performance,analyze \
  --ref file:./performance-benchmarks.json \
  --ref p42:/commands/system-metrics \
  --ref search:"performance optimization patterns" \
  --prompt "Create a performance monitoring tool that tracks application metrics, identifies bo

# Usage
perf-monitor --app myapp --duration 1h --alert-threshold 95th_percentile
```

AI-Assisted Conversations

Project Planning with Context

```
# Strategic project conversation with full context
port42 possess @ai-founder \
  --ref file:./business-requirements.md \
  --ref file:./technical-constraints.md \
  --ref p42:/memory/strategy-session-2024 \
  --ref url:https://industry-report.com/trends \
  "Help me create a 6-month product roadmap that balances business goals with technical feasibi
```

Code Review with AI

```
# Technical code review with architectural context
port42 possess @ai-engineer \
  --ref file:./pull-request-diff.patch \
  --ref file:./architecture-docs.md \
  --ref p42:/commands/code-standards \
  --ref search:"best practices microservices" \
  "Review this code change for security, performance, and architectural alignment"
```

Creative Content with Brand Guidelines

```
# Brand-consistent content creation
port42 possess @ai-muse \
  --ref file:./brand-guidelines.pdf \
  --ref file:./previous-campaigns.md \
  --ref url:https://company.com/style-guide \
  --ref search:"successful marketing campaigns" \
  "Create a product launch campaign that aligns with our brand voice and targets our key demog
```

Advanced Integration Patterns

Multi-Tool Workflow Pipeline

```
# Step 1: Create data extractor
port42 declare tool data-extractor --transforms extract,api,cache \
  --ref file:./api-endpoints.json \
  --ref file:./data-schema.json \
  --prompt "Extract data from multiple APIs with rate limiting, caching, and error recovery"

# Step 2: Create data processor (references first tool)
port42 declare tool data-processor --transforms process,validate,transform \
  --ref p42:/commands/data-extractor \
  --ref file:./business-rules.json \
  --prompt "Process extracted data according to business rules with validation and enrichment"

# Step 3: Create data publisher (references both tools)
port42 declare tool data-publisher --transforms publish,notify,dashboard \
  --ref p42:/commands/data-extractor \
  --ref p42:/commands/data-processor \
  --ref file:./notification-config.json \
  --prompt "Publish processed data to multiple destinations with notifications and monitoring"

# Usage: Full pipeline
data-extractor --config prod | data-processor --rules business | data-publisher --destinations
```

AI-Enhanced Development Workflow

```
# Start with requirements analysis
port42 possess @ai-engineer \
  --ref file:./requirements.md \
  --ref file:./existing-codebase/ \
  --ref p42:/commands/architecture-analyzer \
  "Analyze these requirements and suggest an implementation approach"

# Generate scaffolding based on conversation
port42 declare tool project-scaffolder --transforms generate,scaffold,structure \
  --ref p42:/memory/requirements-analysis-session \
  --ref file:./coding-standards.md \
  --ref p42:/commands/template-generator \
  --prompt "Generate project scaffolding based on the requirements analysis conversation with p

# Create testing framework
port42 declare tool test-generator --transforms test,generate,coverage \
  --ref p42:/commands/project-scaffolder \
  --ref file:./test-patterns.md \
  --prompt "Generate comprehensive test suites including unit tests, integration tests, and end

# Usage: AI-guided development
project-scaffolder --based-on requirements-analysis
test-generator --coverage 90 --patterns bdd
```

Cross-Reference Knowledge Base

```
# Create a knowledge aggregator that references multiple sources
port42 declare tool knowledge-aggregator --transforms aggregate,search,synthesize \
  --ref p42:/memory/all-sessions \
  --ref p42:/commands/documentation-tools \
  --ref p42:/artifacts/all-specs \
  --ref search:"architectural decisions" \
  --ref file:./team-knowledge.md \
  --prompt "Create a knowledge aggregator that can search across all our conversations, documen

# Usage
knowledge-aggregator "What are our established patterns for error handling in microservices?"
knowledge-aggregator "Show me all tools related to data processing with their relationships"
```

Best Practices

Effective Prompt Writing

Good Prompts: - Specific and detailed: "Create a tool that analyzes Nginx logs for security threats and performance issues" - Include requirements: "Build a validator with detailed error messages and auto-fixing capabilities"
- Specify output format: "Generate TypeScript interfaces with JSDoc comments"

Avoid: - Vague requests: "Make a tool for logs" - Missing context: "Build an analyzer" (analyzer for what?) - No quality criteria: "Create documentation" (what kind, for whom?)

Smart Reference Usage

Strategic Combinations:

```
# Layer references from general to specific
--ref url:https://standards.org/spec       # Industry standards
--ref p42:/commands/base-implementation       # Existing tools
--ref file:./project-specific-config.json # Local context
--ref search:"lessons learned"             # Historical knowledge
```

Reference Evolution:

```
# Start simple
port42 declare tool basic-parser --ref file:./data.json

# Evolve with more context
port42 declare tool enhanced-parser \
  --ref p42:/commands/basic-parser \
  --ref file:./complex-schema.json \
  --ref search:"parsing best practices"
```

Workflow Integration

Incremental Enhancement: 1. Start with basic tools using simple references 2. Add AI conversations to explore possibilities
3. Create enhanced tools referencing conversations 4. Build workflows connecting related tools 5. Document patterns for future reuse

---

Next Steps

Ready to create your own context-aware tools? Start with:

1. **Identify your use case** from the examples above
2. **Gather your references** (files, existing tools, documentation)
3. **Write a specific prompt** describing exactly what you want
4. **Test incrementally** - start simple and add complexity
5. **Reference your successes** in future tool creation

For more advanced usage, explore: - Port 42 CLI Reference - Universal Reference System Deep Dive
- AI Agent Conversations Guide - Reality Compiler Architecture

---

Generated with Port 42 Reality Compiler - where thoughts become reality