Indicator Extractor CLI

A Node.js command line application that processes JPEG 1 and PNG files and outputs structured JSON results - either simple information, or Trust Indicator Sets, as defined in ISO 21617-1.

Features

- 📁 **File Processing**: Process files and extract useful information, including as a Trust Indicator Set
- 📊 **Basic Content Analysis**: Provides some useful information about the file and its content
- 🎯 **JSON Output**: Generate structured JSON in various syntaxes
- 🎨 **Pretty Printing**: Support for both minified and pretty-printed JSON output
- 📂 **Directory Management**: Automatically creates output directories as needed
- ✅ **Error Handling**: Graceful error handling with informative messages
- 🧪 **Well Tested**: Comprehensive test suite with Jest
- 🔍 **Code Quality**: ESLint configuration for consistent code style

Installation

```
# Clone the repository
git clone https://gitlab.com/your-username/indicator-extractor.git
cd indicator-extractor

# Install dependencies
npm install

# Install globally (optional)
npm install -g .
```

Usage

Basic Usage

```
# Process a file with minified JSON output
indicator-extractor input.jpg ./output

# Process a file with pretty-printed JSON output
indicator-extractor input.jpg ./output --pretty
```

Using the CLI directly

```
# Run from the project directory
node bin/cli.js input.jpg ./output

# With pretty printing
node bin/cli.js input.jpg ./output --pretty
```

Command Line Arguments

- `<input-file>`: Path to the input file to process (required)
- `<output-dir>`: Directory where the JSON output file will be created (required)
- `-p, --pretty`: Pretty print the JSON output (optional)
- `-s, --set`: Generate a Trust Indicator Set (optional, default is basic content analysis)

Output Format

The CLI generates a JSON file with the following structure:

```
{
  "metadata": {
    "inputFile": "/absolute/path/to/input.txt",
    "fileName": "input.txt",
    "fileSize": 1234,
    "processedAt": "2025-06-20T10:30:00.000Z",
    "fileExtension": ".txt"
  },
  "content": {
    "rawContent": "file content here...",
    "lineCount": 15,
    "characterCount": 1234,
    "wordCount": 200
  },
  "processing": {
    "status": "completed",
    "version": "1.0.0"
  }
}
```

Development

Scripts

```
# Run the CLI
npm start

# Run tests
npm test
```

```
# Run tests with coverage
npm run test:coverage

# Run tests in watch mode
npm run test:watch

# Lint code
npm run lint

# Fix linting issues
npm run lint:fix
```

Testing

The project includes comprehensive tests using Jest:

- **Unit Tests**: Test individual utility functions
- **Integration Tests**: Test the complete CLI workflow
- **Error Handling Tests**: Verify graceful error handling
- **C2PA Tests**: Verify processing of the Content Credentials & JPEG Trust Manifests

```
# Run all tests
npm test

# Run tests with coverage report
npm run test:coverage

# Run tests in watch mode for development
npm run test:watch
```

Code Quality

The project uses ESLint for code quality and consistency:

```
# Check for linting issues
npm run lint

# Automatically fix linting issues
npm run lint:fix
```

ESLint Configuration

The project uses modern ESLint configuration with the following features:

- **Modern JavaScript**: ES2020 support with async/await
- **Node.js Environment**: Configured for Node.js development
- **Strict Rules**: Enforces consistent code style and best practices
- **Jest Support**: Configured for Jest testing environment

Project Structure

```
indicator-extractor/
├── bin/
│   └── cli.js              # Main CLI script
│   └── indicatorSet.js     # Create the Trust Indicator Set
│   └── processManifest.js  # Process any C2PA/JPEG Trust
Manifests
├── tests/
│   ├── utils.test.js       # Utility function tests
│   ├── test-helpers.js     # Utility routines for tests
│   ├── c2pa-processing.test.js         # Tests for C2PA
processing
│   ├── setup.js            # Jest setup
├── output/                 # Output directory for processed files
├── coverage/               # Coverage reports (generated)
├── .github/
│   └── copilot-instructions.md # Copilot custom instructions
├── .gitignore              # Git ignore rules
├── eslint.config.js        # ESLint configuration
├── jest.config.js          # Jest configuration
├── package.json            # Project configuration
├── README.md               # This file
└── LICENSE                 # Project license
```

Configuration Files

Jest Configuration (`jest.config.js`)

- **Test Environment**: Node.js
- **Coverage Collection**: Collects coverage from `bin/**/*.js`
- **Test Pattern**: Matches `**/tests/**/*.test.js`
- **Coverage Reports**: Text, LCOV, and HTML formats

ESLint Configuration (`eslint.config.js`)

- **Modern Config**: Uses the new flat config format
- **Node.js Rules**: Optimized for Node.js development
- **Jest Support**: Includes Jest globals for test files
- **Strict Standards**: Enforces code quality and consistency

Contributing

1. Fork the repository
2. Create a feature branch (`git checkout -b feature/amazing-feature`)
3. Make your changes
4. Run tests (`npm test`)
5. Run linting (`npm run lint`)
6. Commit your changes (`git commit -m 'Add amazing feature'`)

7. Push to the branch (`git push origin feature/amazing-feature`)
8. Open a Pull Request

License

This project is licensed under the ISC License - see the LICENSE file for details.

Changelog

v1.0.0

- Initial release
- Comprehensive test suite
- ESLint integration
- Jest testing framework
- Pretty printing support
- Error handling and validation