JPEG Trust Evaluator

This project is a command line tool that evaluates a JSON file according to the rules defined in a JPEG Trust Profile. It utilizes json-formula rules to assess the validity and compliance of the JSON data.

Table of Contents

Installation

1. Clone the repository:

   ```
   git clone https://github.com/yourusername/jpeg-trust-
   evaluator.git
   ```

2. Navigate to the project directory:

   ```
   cd jpeg-trust-evaluator
   ```

3. Install the dependencies:

   ```
   npm install
   ```

Usage

To run the tool, use the following command:

```
node src/index.js [options] <jsonFile>
```

Required Arguments

- `<jsonFile>` - Path to the JSON file containing JPEG Trust Indicator Sets data to evaluate

Required Options

- `-p, --profile <path>` - Path to the JPEG Trust Profile file (JSON or YAML format)

Optional Options

- `-o, --output <directory>` - Output directory for reports (if not specified, results are printed to console)
- `-y, --yaml` - Output report in YAML format (default is JSON)
- `--html <path>` - Path to HTML template file for generating HTML reports
- `-h, --help` - Display help information
- `-V, --version` - Display version number

Examples

1. **Basic evaluation** (output to console):

   ```
   node src/index.js -p testfiles/camera_profile.yml testfiles/camera_indicators.json
   ```

2. **Generate JSON report** in output directory:

   ```
   node src/index.js -p testfiles/genai_profile.yml -o output testfiles/genai_indicators.json
   ```

3. **Generate YAML report**:

   ```
   node src/index.js -p testfiles/no_manifests_profile.yml -o output --yaml testfiles/no_manifests_indicators.json
   ```

4. **Generate HTML report** using a template:

   ```
   node src/index.js -p testfiles/camera_profile.yml -o output --html testfiles/report_template.html testfiles/camera_indicators.json
   ```

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

Contributing

Contributions are welcome! Please open an issue or submit a pull request for any enhancements or bug fixes.

Submitting a Pull Request

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 MIT License. See the LICENSE file for more details.

Changelog

v1.0.0

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