Validating FedRAMP Content with OSCAL CLI

Get Started

The oscal-cli is an open source command-line utility designed to help developers and security professionals interact with OSCAL. To get started, follow the installation instructions from the OSCAL-CLI GitHub "README" page. Once installed, you can use the tool to perform core tasks such as validating OSCAL JSON, YAML, or XML files against the official schemas, converting between formats, and resolving "profile" documents into comprehensive "catalog" views. For those working in containerized environments, the tool is also available as a Docker image, allowing for easy integration into automated CI/CD compliance pipelines.

Commands

The following are a brief summary of the most commonly used oscal-cli commands when working with FedRAMP OSCAL documents.

Validate

Checks that the specified OSCAL file is well-formed and valid per OSCAL model and any additional specified constraints.

```
oscal-cli validate <file> -c <constraint-file> [options]
```

Convert

Converts an OSCAL source file to the specified file format (e.g., XML, JSON, or YAML).

```
oscal-cli convert --to=FORMAT <source-file-or-URL> <destination-file>
```

Profile Resolution

Creates a new tailored OSCAL catalog by combining baseline imported controls and applying any modifications, as specified in the source profile.

```
oscal-cli resolve-profile <profile-file>
```

Metaschema Validate

Validates that a Metaschema file is well-formed and valid per the specified constraint definitions.

```
oscal-cli metaschema validate <file> -c <style-guide> [options]
```

FedRAMP Validation Examples

Validating SSP Against FedRAMP Constraints

```
oscal-cli validate path/to/ssp.xml \
  -c path/to/fedramp-external-allowed-values.xml \
  -c path/to/fedramp-external-constraints.xml
```

Understanding Validation Errors

Common Error Messages

1. UUID Warning:

```
[WARNING] [.../protocol[1]] It is a best practice to provide a UUID.
```

Resolution: Add a unique UUID to the specified element.

2. Constraint Violations:

- Invalid allowed values
- Missing required fields
- Pattern matching failures

SARIF Output Integration

SARIF (Static Analysis Results Interchange Format) provides detailed validation results viewable in VS Code.

Generate SARIF Output

```
oscal-cli validate path/to/file.xml \
  -c path/to/constraints.xml \
  --sarif-include-pass \
  -o results.sarif.json
```

SARIF Features

- Detailed error locations
- Stack traces for debugging
- Pass/fail results for each rule
- VS Code integration for visual feedback

Best Practices

1. Always validate at both levels:
   - Basic OSCAL schema validation
   - FedRAMP constraints validation
2. Always validate against both allowed values and external constraints
3. Use SARIF output for detailed analysis:

   ```
   oscal-cli validate path/to/document.xml \
     -c path/to/constraints.xml \
     --sarif-include-pass \
     -o results.sarif.json
   ```

4. Address all warnings, even if they don't cause validation failures
5. Keep constraint files updated with latest FedRAMP requirements

---

Revision #4
Created 2026-04-07 14:53:16 UTC by Rene M. Tshiteya
Updated 2026-04-07 15:50:07 UTC by Rene M. Tshiteya