FedRAMP

OSCAL Developer Data Bites

March 7, 2024

info@fedramp.gov

fedramp.gov

Introduction

Purpose: To create space for dialogue between developers who use OSCAL and the FedRAMP® automation team.

Outcomes:

- Shared understanding of the projected future of schematron and using metaschema validation mechanisms via OSCAL-CLI tool
- Shared understanding of the namespace collisions on validations
- Productive discussion around OSCAL

Agenda:

- Welcome
- General Updates
- FedRAMP Automation Community Updates
- Pre-Submitted Q&A
- Future of schematron and using metaschema validation mechanisms via OSCAL-CLI tool
- NIST OSCAL vs. FedRAMP Namespace collisions on validations
- Open Forum
- Next Steps & Closing

Data Bites Guiding Principles

- Keep the discussion respectful
- Be curious, seek understanding
- Speak from your own experience
- Challenge through questions
- Focus on ideas
- Keep it technical

General Updates

FedRAMP Automation Community Updates

March 7, 2024

Revising OSCAL Guides

- FedRAMP automation team is continuing to work towards publishing HTML versions of the OSCAL guides to replace the current PDF versions.

Local Validation Tooling

- FedRAMP automation team is working on adding metaschema validation mechanisms in the OSCAL-CLI tool

GitHub Issues

- Prioritizing issues related to FedRAMP Guides and SP 800-53 Rev 5
  - Issues #555, #558, #563, #534, and #556

Closed PRs

- #540 Local version of SP 800-53 with zero padded labels; updated based on usnistgov/oscal-content#238
- #557 Container support for user guides

Pre-Submitted Questions

Pre-Submitted Questions

None received!

Reminder to submit questions/topic ideas via **https://forms.gle/M4pT7P2xyE6hRC7DA**

Migration from Schematron to Metaschema Validation Mechanisms via OSCAL-CLI Tool

FedRAMP OSCAL Validation Goals

Provide a means to **validate FedRAMP OSCAL packages before submission** to FedRAMP for **completeness**, **accuracy**, and to ensure the package is **free of errors.**

Our goals:

- Define fully how to use OSCAL to represent a FedRAMP package.
- Help creators of OSCAL packages ensure all OSCAL and FedRAMP specific requirements are met.
    - **Completeness:** Ensure that required content is provided.
    - **Consistency:** Normalize package data to **enable machine analysis**.
    - **Free from Error:** Find common data errors (e.g., broken cross-references, invalid/nonsensical values) before submission.
- Increase consumer confidence in FedRAMP OSCAL packages to improve the consumer experience and reduce review times.

Today's discussion will focus on how to achieve these goals.

OSCAL Validation Tooling - Current State

FedRAMP Schematron Validations

This solution has many limitations:

- XML only validation; no support for JSON or YAML
- Out of sync with latest FedRAMP OSCAL guidance
- Many false positives
- Complex to run and maintain

OSCAL Command Line (OSCAL-CLI) Validation tool

This solution supports core OSCAL validation:

- Supports JSON, YAML, and XML
- Supports core OSCAL validation rules
- No current support for FedRAMP-specific requirements
- Integrates easily into CI/CD pipelines

Both tools are currently used for FedRAMP OSCAL validation.

FedRAMP Validation Resources - Current State

FedRAMP Validation Resources are scattered across the repository

Managing these resources is difficult:

- Multiple resources to update, leading to drift, errors, and inconsistencies
- Machine-readable requirement content is decentralized and difficult to consume/understand

Schematron rules:

https://github.com/GSA/fedramp-automation/tree/master/src/validations/rules

Extensions:

https://github.com/GSA/fedramp-automation/blob/master/dist/content/rev5/resources/xml/FedRAMP_extensions.xml

Allowed values:

https://github.com/GSA/fedramp-automation/blob/master/dist/content/rev5/resources/xml/fedramp_values.xml

FedRAMP OSCAL Validation - Future State

Provide a ***single validation platform*** that supports validation of FedRAMP OSCAL packages against the **combination** of **core OSCAL** and **FedRAMP-specific requirements**

Our objectives:

- Support local validation of FedRAMP packages prior to submission to FedRAMP
- Synchronize FedRAMP OSCAL guides and validations
- Support all OSCAL formats, e.g., JSON, YAML, XML
- Ensure that all requirements are unit tested to ensure validations are correct and maintainable as OSCAL and FedRAMP guidance changes
- Enable collaboration on maintaining validation rules and tests through use of GitHub

Transitioning to Metaschema-Based FedRAMP Validations

FedRAMP plans to use the OSCAL CLI for both core OSCAL and FedRAMP validations

- Provides a single set of validation results for OSCAL and FedRAMP requirements
- Provides a common platform that can be used in multiple modes: CLI (existing), and GUI or REST API (future)
- FedRAMP extensions will be defined using a single external Metaschema constraints
  - Replaces the current multiple files used to define rules, extension, allowed values
  - Provides a foundation on which other parties can define their own additional constraints

Would it be useful to release a distinct FedRAMP version of the OSCAL CLI tool with the extra FedRAMP validations bundled?

Or would it be useful to have a “FedRAMP” mode in OSCAL CLI?

Development Prioritization

Rough priorities for development:

1. Establish unit testing framework for continuous unit testing of validations.
2. Identify and fix inconsistencies where the FedRAMP OSCAL guides and validation rules differ.
3. Identify and fix false positives/negatives in validation results.
4. Validation rule enhancements to ensure required information is provided to improve reviewer experience.
5. Validation rule enhancements to support additional data needed by FedRAMP and FedRAMP stakeholders.

Development Approach

Analyze Requirements Mismatches

Launch automation website and OSCAL markdown-based guides.

Identify requirement inconsistencies and gaps in current Schematron validations.

Begin guide improvements.

Develop OSCAL-CLI-Based Tooling

Develop initial FedRAMP Metaschema constraints

Release initial open source OSCAL validation tooling for alpha testing

Continue OSCAL guide improvements.

Continued Refinement

Continue to improve validations and guides.

Multiple validation tooling releases for beta testing.

Transition to regular maintenance of OSCAL requirements, guides, and validations.

NIST OSCAL vs. FedRAMP Namespace Collisions on Validations

NIST OSCAL vs. FedRAMP Namespace Collisions

Background

- FedRAMP defined many extensions to support its specific use cases and data needs
  - Many of these extensions were created for Rev 4 and predate more recent versions NIST OSCAL which include similar props
  - More recent extensions were added to support Rev 5
  - This has led to cases where extensions have namespace collisions
- Need a comprehensive review of existing FedRAMP extensions, clear and consistent guidance around when to use the extensions, and closer alignment with core OSCAL

NIST OSCAL vs. FedRAMP Namespace Collisions

Approach for Resolving Namespace Collisions

Each FedRAMP extension will be reviewed and considered. There are 4 possible resolutions:

1. Keep extension as-is (do nothing)
2. Deprecate the extension that is no longer needed

   FedRAMP no longer requires the information
3. Transition to core OSCAL approach; deprecate FedRAMP extension

   OSCAL has already generalized the case
4. Propose new OSCAL allowed value(s); deprecate FedRAMP extension

   Useful where a generalizable case exists

In all cases, the OSCAL guides need to be checked for accuracy and completeness.

Example 1 - Deprecate the extension

- Prior versions of the FedRAMP SSP template had counts of the following user types, however, FedRAMP no longer requests this information:
  - **users-internal** - a current number of users internal to the organization
  - **users-external** - a current number of users external to the organization
  - **users-internal-future** - the anticipated number of users internal to the organization in one year
  - **users-external-future** - the anticipated number of users external to the organization in one year

- Other examples include privacy related extensions:
  - **privacy-designation** - indicates whether this system is privacy sensitive
  - **privacy-threshold-analysis-q#** - Privacy Threshold Analysis questions

Example 2 - Transition to core OSCAL approach

- FedRAMP has an **interconnection-direction** OSCAL extension
  - For components of type “interconnection”, this extension prop identifies the direction of information flow for the interconnection
  - core OSCAL provides native support for this information via its **direction** prop

- FedRAMP has a **raw-tool-output** allowed value extension on back-matter resource “type” props
  - core OSCAL provides native support for this via its defined **tool-output** and **raw-data** allowed values for back-matter resource “type” props

Example 3 - Improve core OSCAL approach

- FedRAMP has an **authentication-method** extension, used to indicate the authentication method(s) for users of a leveraged service or external interconnection.
  - Values should specify authentication methods in NIST 800-63B (https://pages.nist.gov/800-63-3/sp800-63b.html)
  - Could be considered for addition to core OSCAL

FedRAMP OSCAL Versioning

To support a transition, there is a need to incrementally change FedRAMP OSCAL requirements

- Changes to requirements will be disruptive to implementers
- Guides and validations need to move as a unit with a given release
- Content and tooling produced needs a clear version target for implementation
- Semantic or schema versioning can be a useful way of signaling compatibility
    - https://semver.org/
    - https://snowplow.io/blog/introducing-schemaver-for-semantic-versioning-of-schemas/

Should this cleanup be done as a single release or through multiple releases?

Any strong opinions around using semantic or schema versioning?

Open Forum

Thank you

Our next Developer Data Bites virtual meeting will be on

Thursday, April 4, 2024 at 12p ET.

Submit questions and future discussion topics to OSCAL@fedramp.gov

Learn more at fedramp.gov

@FEDRAMP

Collaborating with FedRAMP

Collaboration Resources

FedRAMP Automation GitHub: https://github.com/GSA/fedramp-automation

- Open Issues: https://github.com/GSA/fedramp-automation/issues
- Open Pull Requests: https://github.com/GSA/fedramp-automation/pulls
- Active Work: https://github.com/orgs/GSA/projects/25/views/3
- Community Review Needed: https://github.com/orgs/GSA/projects/25/views/7

GitHub Resources:

- Issues: https://docs.github.com/en/issues
- Pull Requests: https://docs.github.com/en/pull-requests

How to Submit Issues with FedRAMP

Ensuring your outstanding issues or questions are received:

Issues can be submitted in several ways:

✓ Preferred	Alternate
Open an issue on fedramp-automation github so that it will benefit the NIST/FedRAMP community. https://github.com/GSA/fedramp-automation/issues	Email us at oscal@fedramp.gov

OSCAL Resources

NIST:
https://pages.nist.gov/OSCAL/
Learning Resources: https://pages.nist.gov/OSCAL/learn/
Current release: https://github.com/usnistgov/OSCAL/releases
Development version: https://github.com/usnistgov/OSCAL/tree/develop
Content repo: https://github.com/usnistgov/oscal-content

FedRAMP:
Current repo: https://github.com/GSA/fedramp-automation
Current issues: https://github.com/GSA/fedramp-automation/issues
Validations work: https://github.com/18F/fedramp-automation/tree/master/src/validations
Web based validation tool:
https://federalist-2372d2fd-fc94-42fe-bcc7-a8af4f664a51.app.cloud.gov/site/18f/fedramp-automation/#/documents/system-security-plan