OSCAL Community Capabilities

NIST OSCAL Mini-Workshop
March 20, 2024

Brian Ruf
Chris Roblee

Community Features

- https://OSCAL.io
- OSCAL Content Registry
- OSCAL Viewer
- OSCAL REST OpenAPI Specification
- Supporting Documentation
- Other Community Resources?

https://OSCAL.io

- A portal for the OSCAL community to get started and share resources
- Events
- Known Tools
- Available Communication Channels (Planned)
- Blog/Article Posting (Planned)
- OSCAL Content Registry
- OSCAL Viewer

Questions or Submissions: oscal@oscal.io

Brief Tour of OSCAL.io

Coming Soon

- Tools List and Communication Channels
- Criteria for inclusion
- Required and optional publication details

- Draft criteria and information requirements – Public comment period
- Any entry meeting the criteria will be included

- Long-Term: Self-service submission and management of entries
- Shorter Term: Send requests to oscal@oscal.io

Tools

Considerations for Criteria

- Must support any version of the OSCAL standard (1.0.0 and later)
- Both licensed and open source (for-profit or free) welcome

Considerations for Tool Details

- Tool name *
- Brief description *
- Version(s) of OSCAL supported *
- Owner (person or org) *
- License Type *
- Contact information (may be sales, support, or other)
- Web site and/or Git Repo

Communication Channels

Considerations for Criteria

- Must be relevant to OSCAL (development, implementation, adoption)
- Do we require the channel to be available to the public?

Considerations for Channel Details

- Channel name *
- Brief description *
- Owner (person or org) *
- Open or Closed Community (and who is eligible if closed) *
- Contact information
- Joining link and/or instructions
- Link to any rules of behavior or governing document

The OSCAL Registry

- Beta testing now
- Production will be available at **https://registry.oscal.io**
- Anyone can view the content anonymously
- Anyone can create an account to post or bookmark content
- Upload valid OSCAL XML, JSON or YAML files
- Currently Supports: Catalogs, Profiles and Component Definitions
- GUI and API enabled: Your tools can integrate with the registry

OSCAL Registry Capabilities

- **Easy viewing, uploading, managing, and sharing** of OSCAL documents.
- Individual user profiles
- Secure, encrypted, cloud-native storage of all documents.
- **Support all official OSCAL releases** (1.0.0 to 1.1.2) and **Catalog**, **Profiles**, and **Component Definition** models.
- Upload/download OSCAL documents in **XML, JSON, or YAML** formats.
- **Fuzzy search** and filtering capabilities.
- **Validation** and conversion of documents up to 10.5MB.
- **Hosted content** including NIST SP 800-53, SP 800-63, Federal PKI, and ISM OSCAL baselines.

Registry Demo

Feedback we've received so far …

- "Just the ability to upload and convert between formats is big deal"
- "I like that you have the like button, Can you add a share button?"
- Suggest including package **download counts**
- Suggest Including **verified publisher option** (blue checkmark?)
- Suggest **File versioning**
- "Can you segment a fragment of a model in a different format?"

What Else?

To provide feedback, send email to oscal@oscal.io

OSCAL REST OpenAPI Specification

- Releasing as an open-source specification
- Built for tool-to-tool and org-to-org hand-off of OSCAL content
- Exchange any OSCAL format plus attachments
- Manage file versioning: create and request snapshots in time
- Syntax provides for profile resolution and resolution snapshots
- Room for implementation-specific details and handling

OSCAL REST OpenAPI Specification

GET, POST	/[model-name]
GET, PUT, PATCH, DELETE	/[model-name]/{identifier}
GET, POST	/[model-name]/{identifier}/snapshot
GET, PUT, PATCH, DELETE	/[model-name]/{identifier}/snapshot/{identifier}
GET, POST	/[model-name]/{identifier}/attachment
GET, PUT, PATCH, DELETE	/[model-name]/{identifier}/attachment/{identifier}

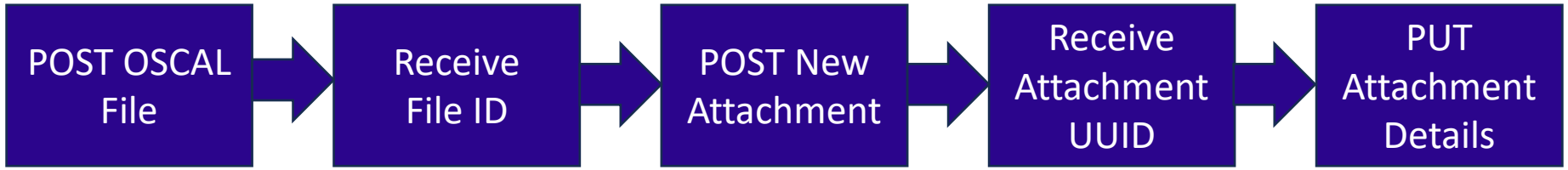
Replace [model-name] with:

- catalog
- profile
- component-definition
- system-security-plan
- assessment plan
- assessment-results
- plan-of-action-and-milestones

OSCAL REST OpenAPI Specification

`GET /system-security-plan` -> list of available SSPs. Includes implementation-assigned identifiers

`GET /system-security-plan/{ssp-id}` -> Desired SSP – Header field specifies format (XML, JSON, YAML)



`POST /system-security-plan` -> implementation-assigned SSP identifier

`POST /system-security-plan/{ssp-id}/attachment` -> implementation-assigned Attachment UUID

NOTE: The implementation receives the file and creates a back-matter resource in the content, including resource UUID and an *rlink* pointing to the file.

`PUT /system-security-plan/{ssp-id}/attachment/{uuid}/resource`

Uses the resource assembly to provide any additional details about the attachment, which are assigned to the back-matter resource associated with the attachment.

OSCAL REST OpenAPI Specification

- Draft 0.2.1 out now for public review and feedback
- https://github.com/EasyDynamics/oscal-rest/wiki/OSCAL-REST-OpenAPI-Specification-Wiki
- https://app.swaggerhub.com/apis-docs/brian-easyd/oscal-rest/0.2.1
- To provide feedback, either send email to oscal@oscal.io or open an issue at: https://github.com/EasyDynamics/oscal-rest/issues

Discussion Topic: OSCAL Extensions

- Have you defined your own OSCAL extensions?
- What OSCAL extensions are relevant to you?
- Are you struggling to manage extensions, allowed values and other constraints outside of the core OSCAL syntax?
- Is there interest in a community-driven standard for capturing extensions, allowed-values and other constraints?

Open Discussion

On the Horizon: **Community Plans**

- **New community capabilities** enabling simplified sharing and discovery of OSCAL artifacts
- **Updates to draft API specification** and how-to's for taking advantage
- **Community-driven content:** Have ideas? We'd love to hear it! Email us at oscal@oscal.io

Thank You!