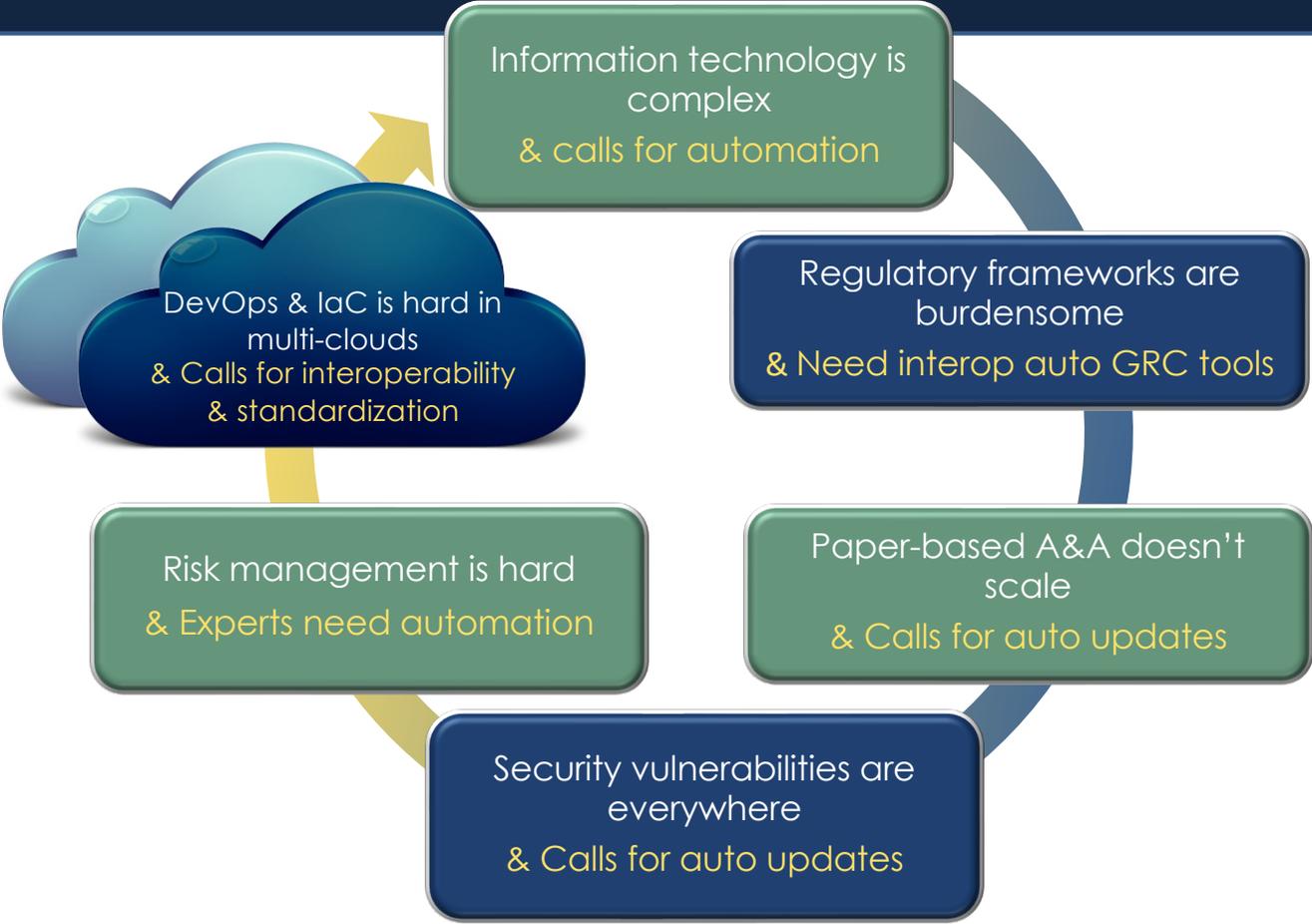


Open Security Controls Assessment Language What is OSCAL and Who Needs It?

Dr. Michaela Iorga,
OSCAL Strategic Outreach Director

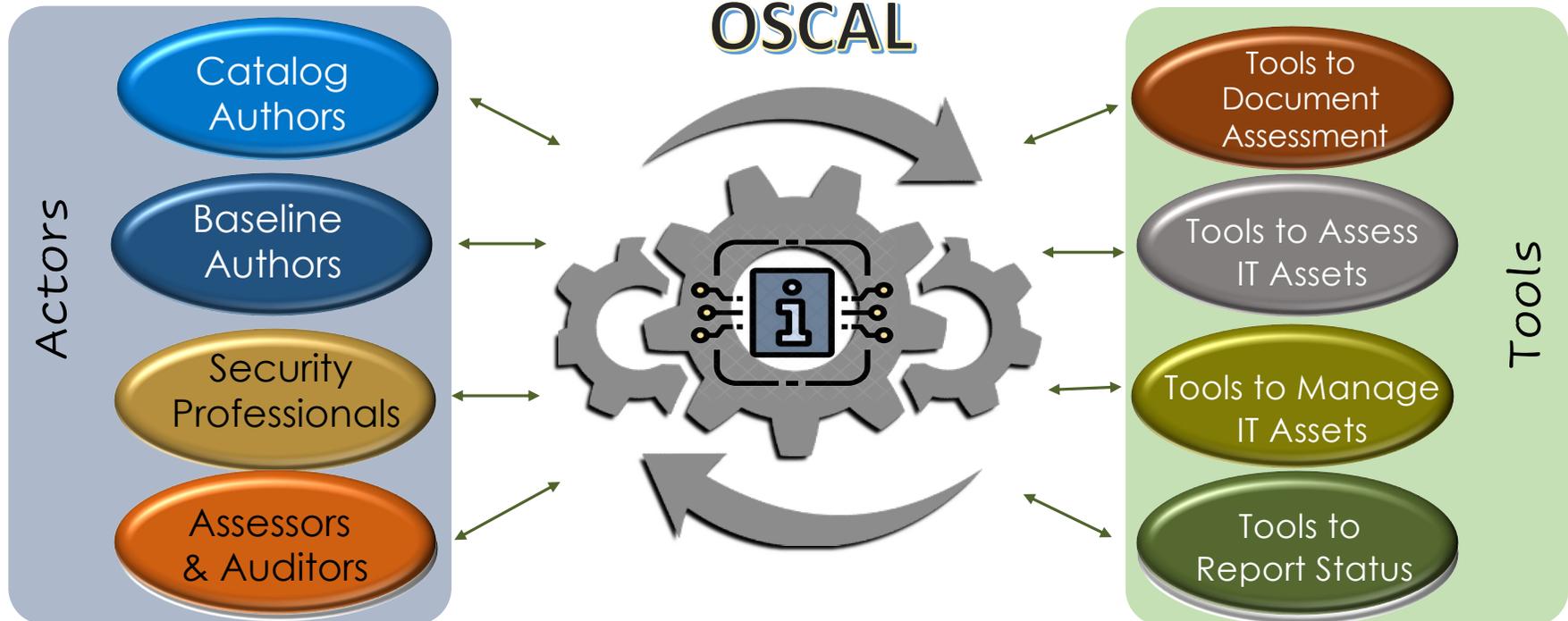
David Waltermire,
OSCAL Technical Director

Why are we all here today?



What was needed?

A (Cyber) Machine-readable Esperanto that enables actors, tools and organizations to exchange information via automation:

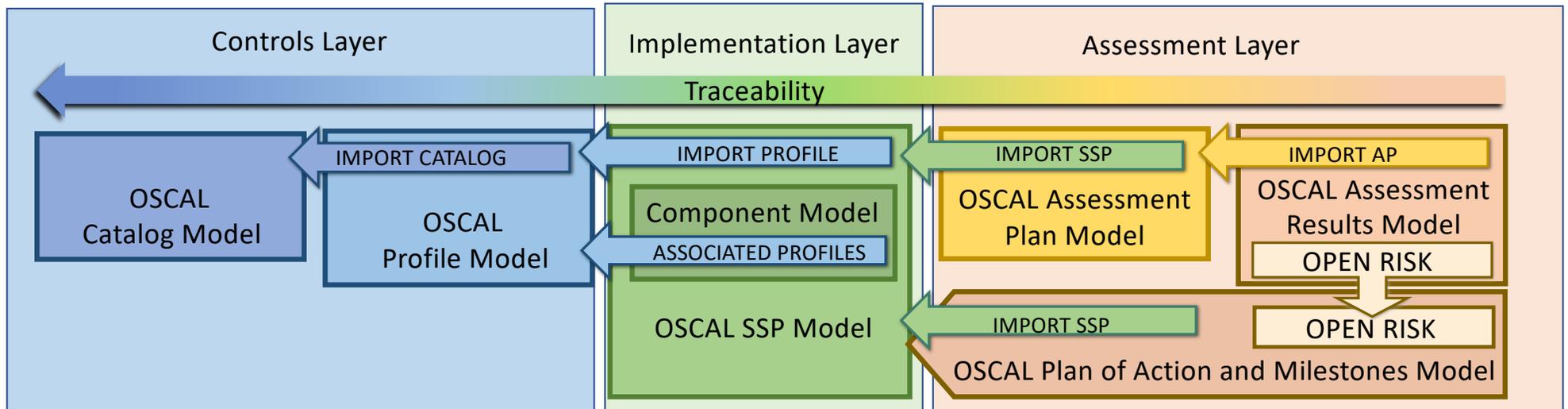


OSCAL sets the foundation for automation and interoperability

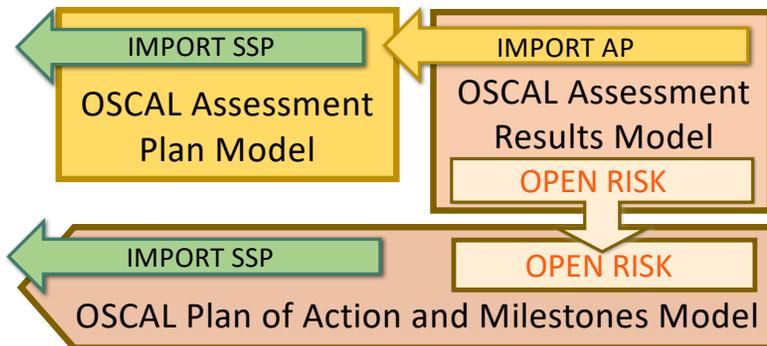
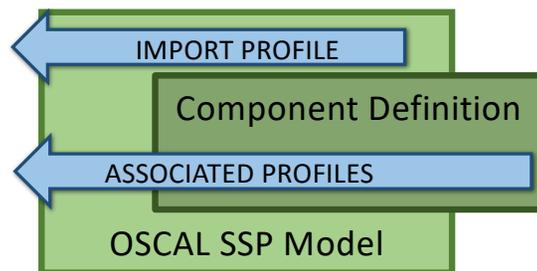
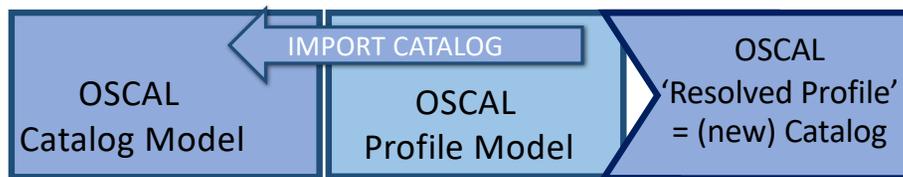
What is OSCAL?

OSCAL is the result of NIST and FedRAMP collaboration

- **OSCAL provides** a **common/single machine-readable language**, expressed in XML, JSON and YAML for:
 - ❑ multiple compliance and risk management frameworks (e.g. SP 800-53, ISO/IEC 27001&2, COBIT 5)
 - ❑ software and service providers to express implementation guidance against security controls (Component definition)
 - ❑ sharing how security controls are implemented (System Security Plans [SSPs])
 - ❑ sharing security assessment plans (System Assessment Plans [SAPs])
 - ❑ sharing security assessment results/reports (System Assessment Results [SARs])
- **OSCAL enables automated traceability** from selection of security controls through implementation and assessment



How is OSCAL different?



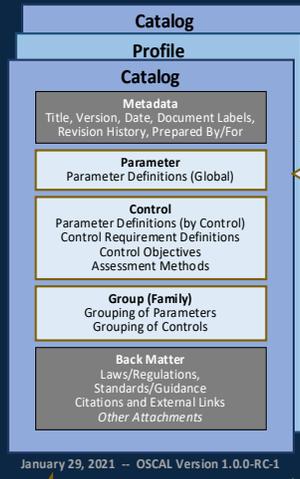
- No information needs duplication
- Custom granularity (controls can be decomposed into statements)
- Unique identifiers for parameters and statements

- Vendors can document their products
- Systems' security implementation can be decomposed

- Capture assessment Plans and Activities with custom cadence, & only for selected components
- POA&M conveys open risks aligned with the SSP capabilities and controls

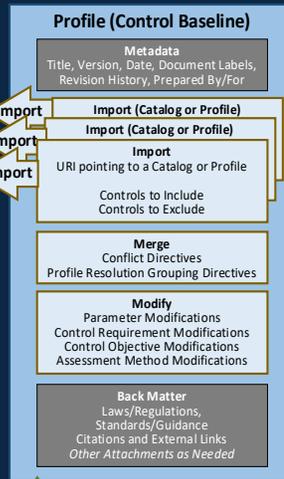
A Closer Look at OSCAL Models

CATALOG MODEL



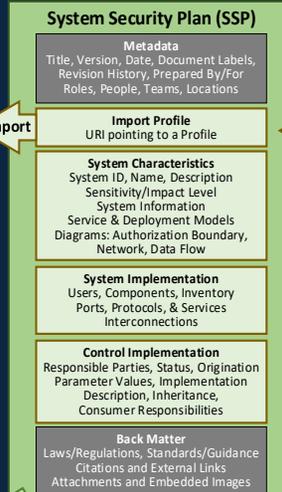
The import arrow identifies what OSCAL content is linked as a result of the import statement. Imported content is referenced, not copied.

PROFILE MODEL



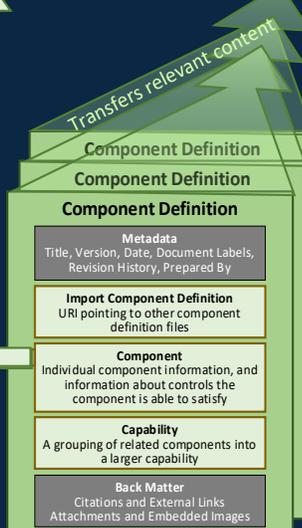
Associates configuration settings with baselines

SSP MODEL

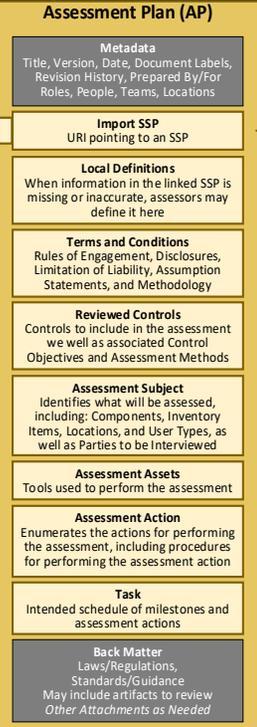
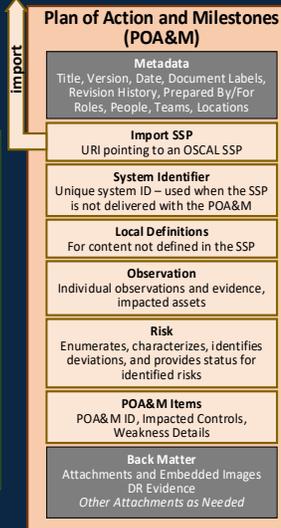


Transfers relevant content

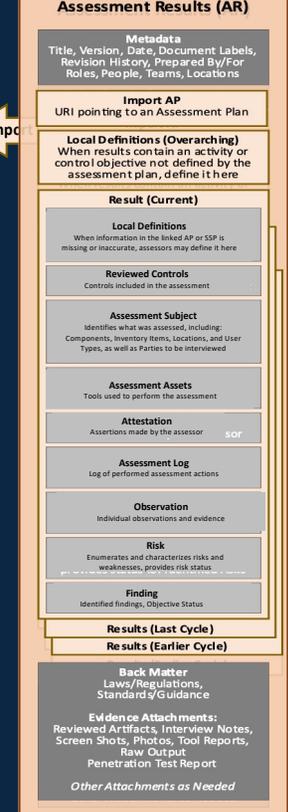
COMPONENT MODEL



POA&M MODEL



ASSESSMENT PLAN MODEL



ASSESSMENT RESULTS MODEL

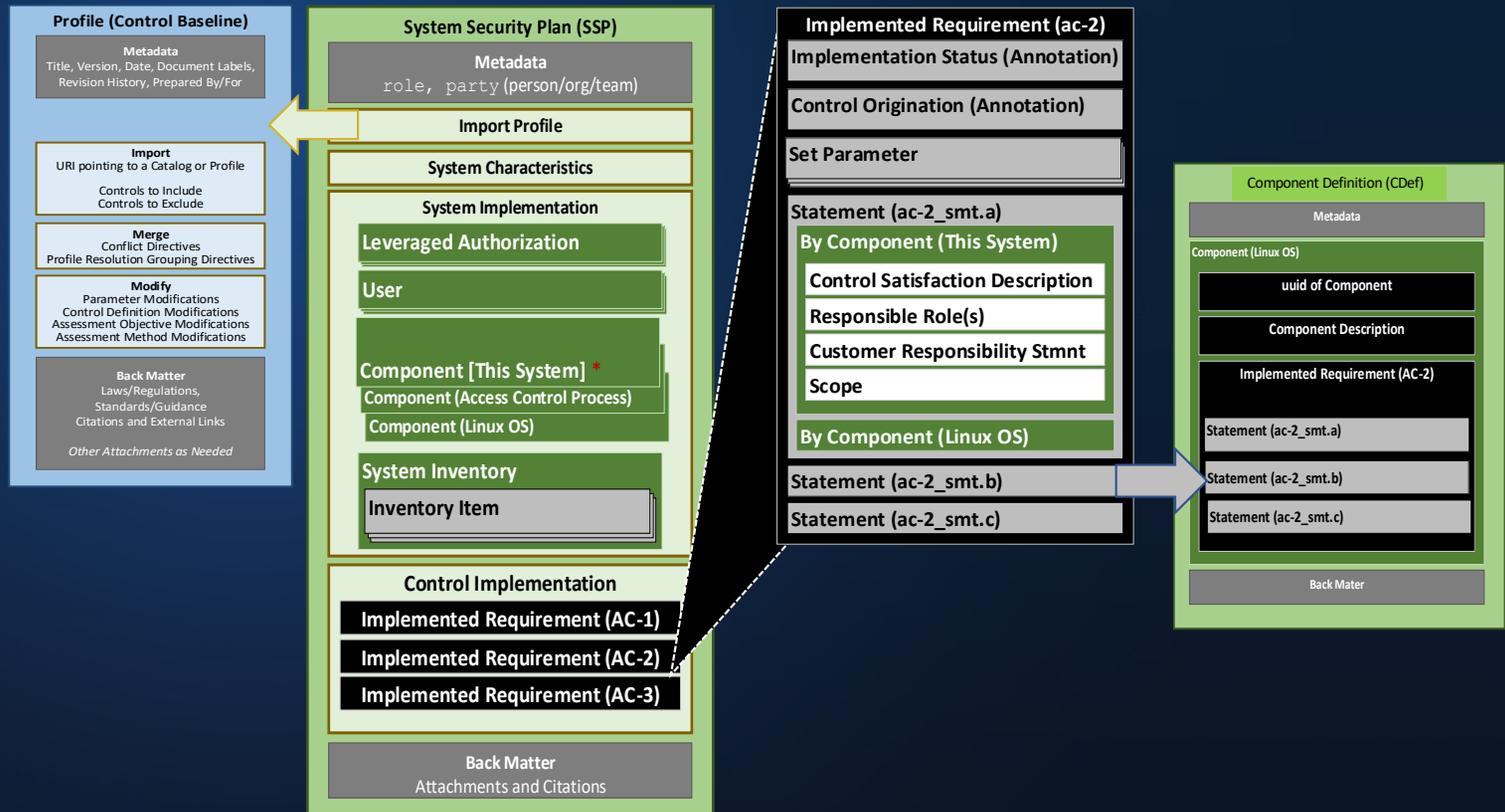


Anatomy of the OSCAL Implementation Layer

Where the Innovation Truly Starts: The OSCAL Implementation Layer

OSCAL SSP:

- Imports a Profile identifying the controls
- Each control response is broken down to the individual components involved.
- Enables a more robust response to controls
- Example: The access control implementation that satisfies AC-2, part a is described separately for:
 - This System
 - The Access Control Procedure
 - A shared Application



* Every SSP, must have a component representing the whole system.



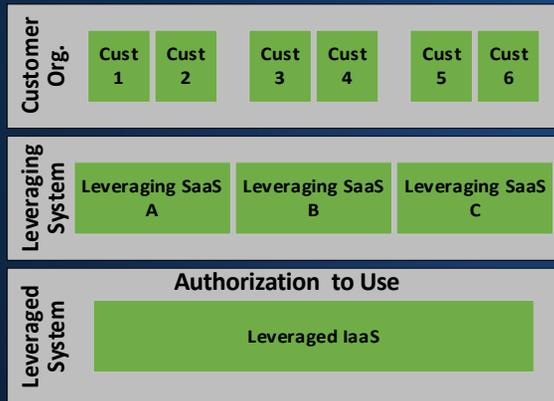
OSCAL Support for Continuously Authorizing Systems to Operate

Authorization to Use Common Control Authorization



Common Control Authorization & Authorization to Use

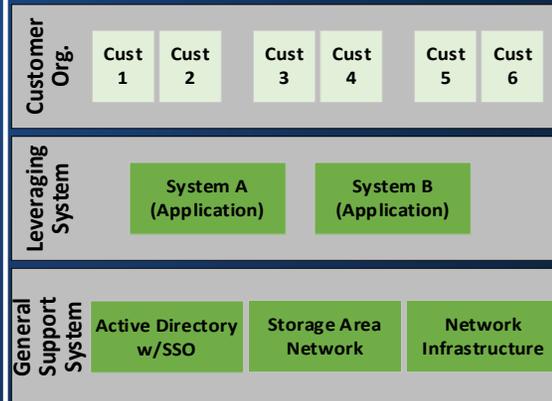
Yes



Cloud (SaaS on IaaS)

Cloud: Several SaaS systems running on a separately authorized IaaS.

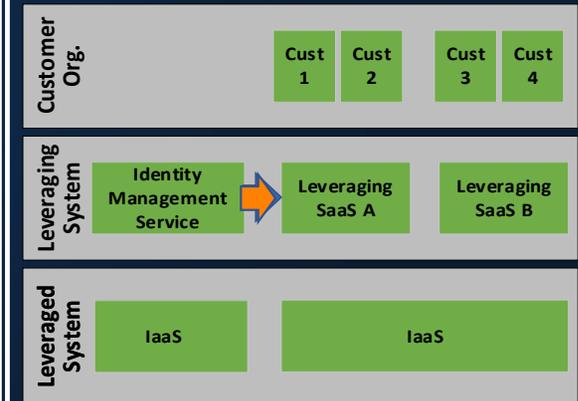
Yes



Data Center (System on GSS)

Data Center: Several systems relying on a separately authorized storage array or other general support system (GSS)

No



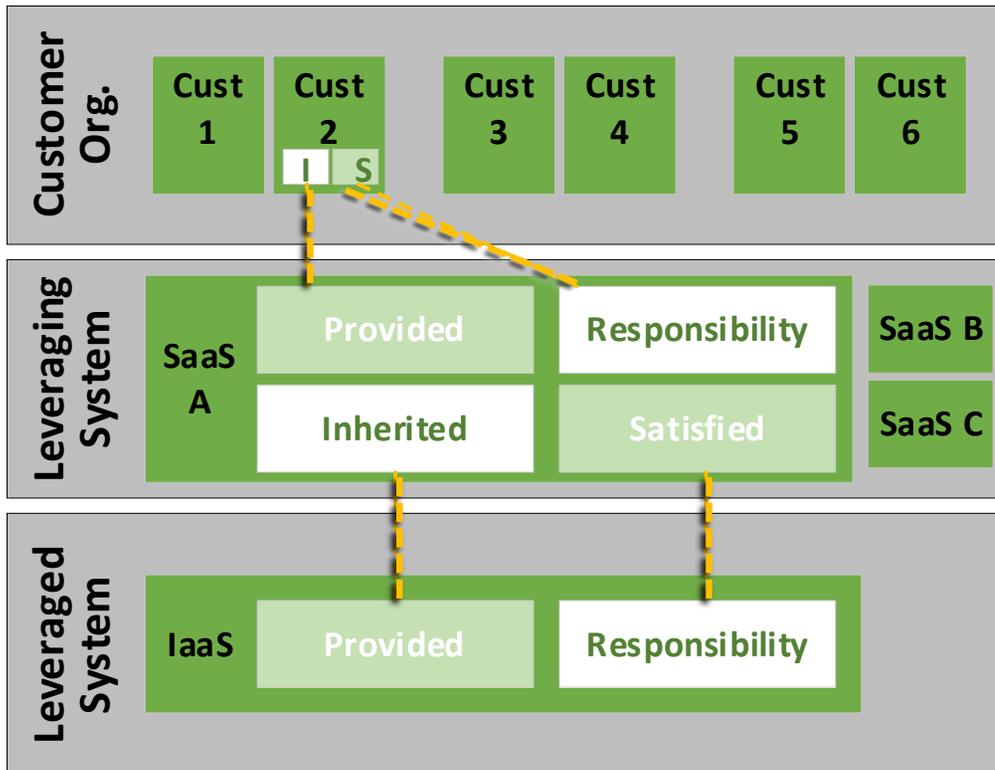
External Service or Interconnection

Interconnections or External Services are not leveraged authorizations

- Even if they have an authorization
- SaaS A handles the Identity Management Service as a system component

OSCAL supports this, just not as a L.A.

OSCAL supports leveraged ATOs of complex stacked systems



Leveraging System:

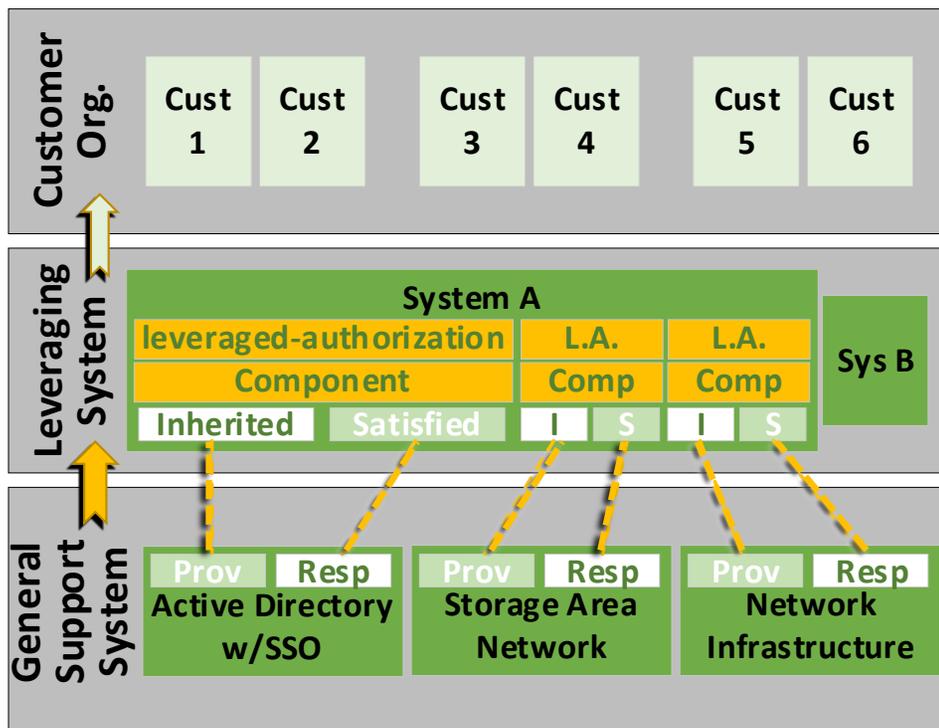
The leveraging system's SSP should:

- identify what is inherited from a leveraged system
- identify any addressed responsibilities (as identified by the leveraged system)

In addition to:

- identifying what **may be** inherited by the leveraging system's customers
- any responsibilities the leveraging system's customers must address to fully satisfy a control

When a Leveraging System has more than one Leveraged System



The same syntax is used

- It is simply replicated for each leveraged system

The Leveraging System's SSP:

- Has a separate "leveraged-authorization" assembly for each leveraged system.
- Has a separate "component" representing each leveraged system.
- Has a separate "component" representing the leveraged system components associated with inherited capabilities.



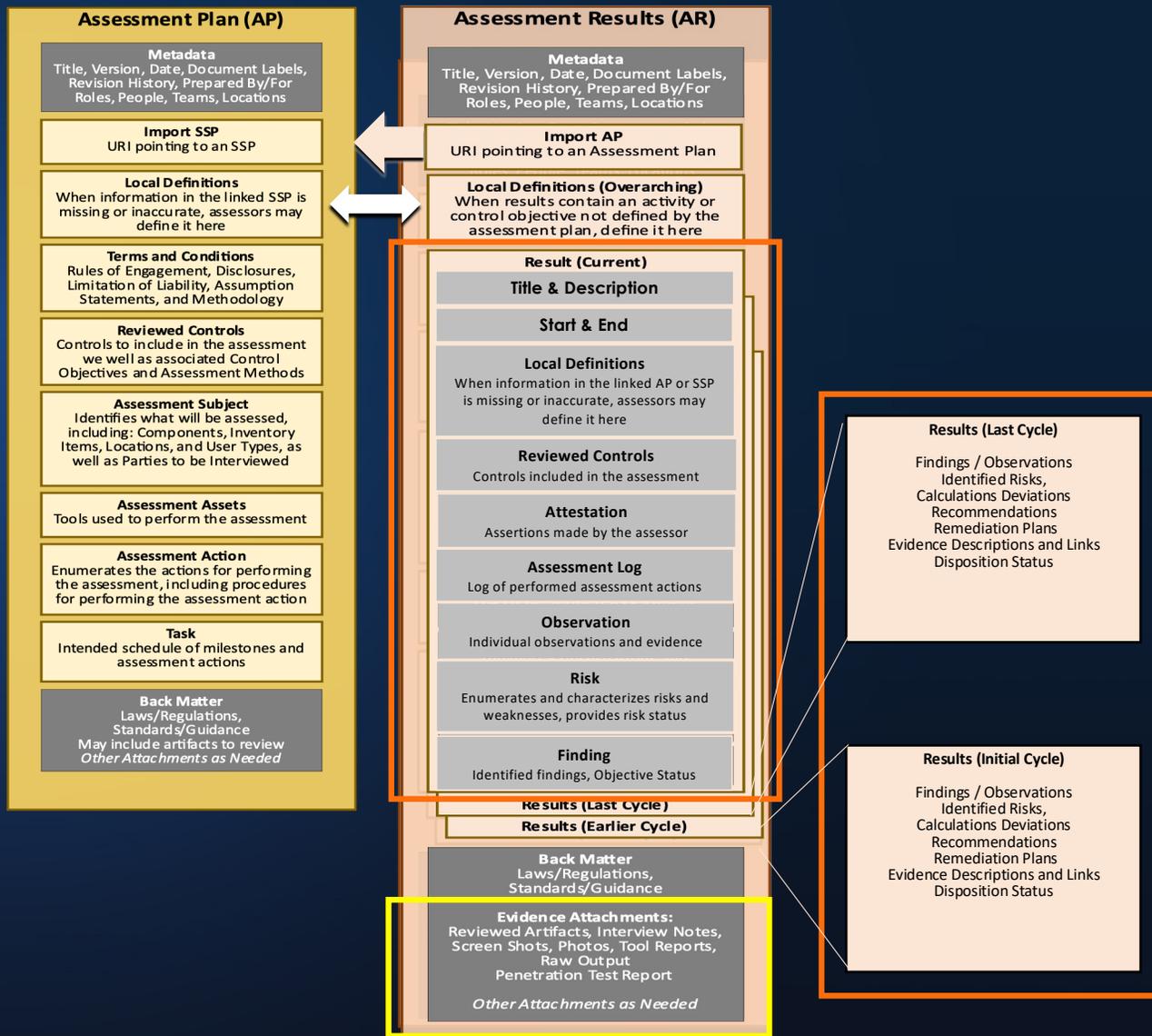
Anatomy of the OSCAL Assessment Layer

Assessment Plan (AP) & Assessment Results (AR)

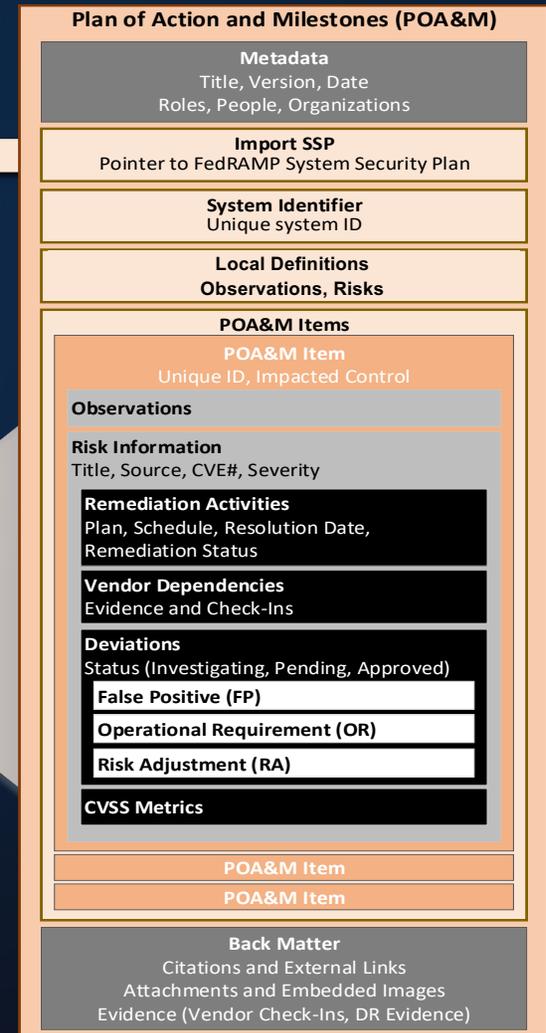
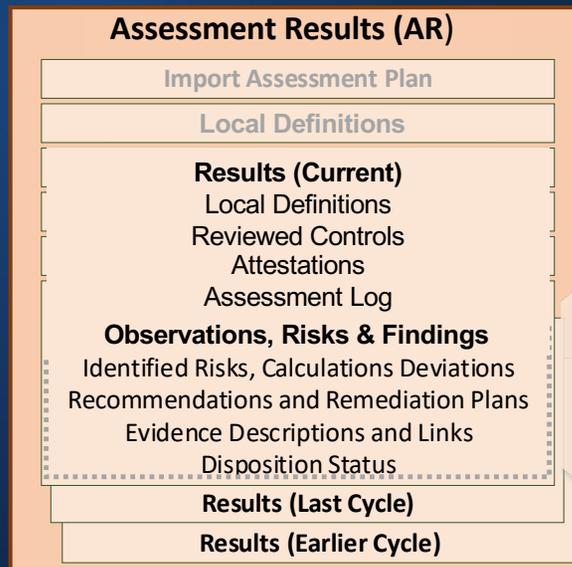
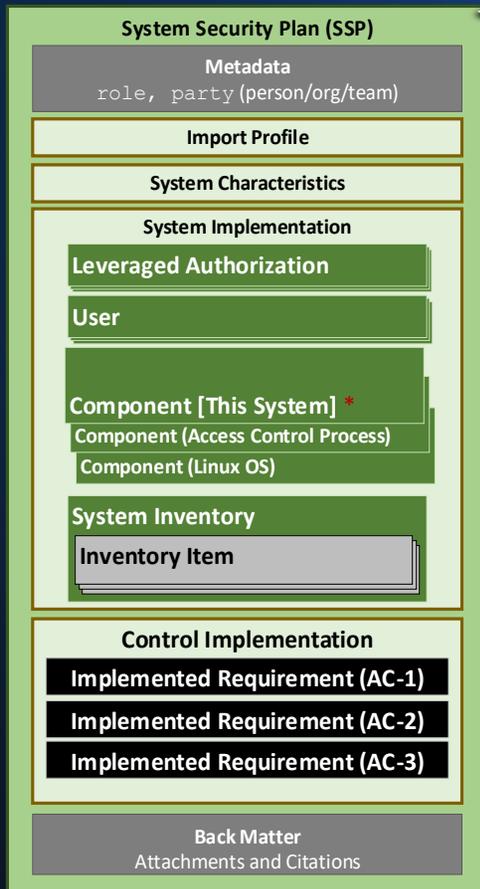
- OVERLAPPING SYNTAX
- SIMILAR BUT DISTINCT PURPOSE
- UNIQUE to AR: **Results** (& **Evidence**)

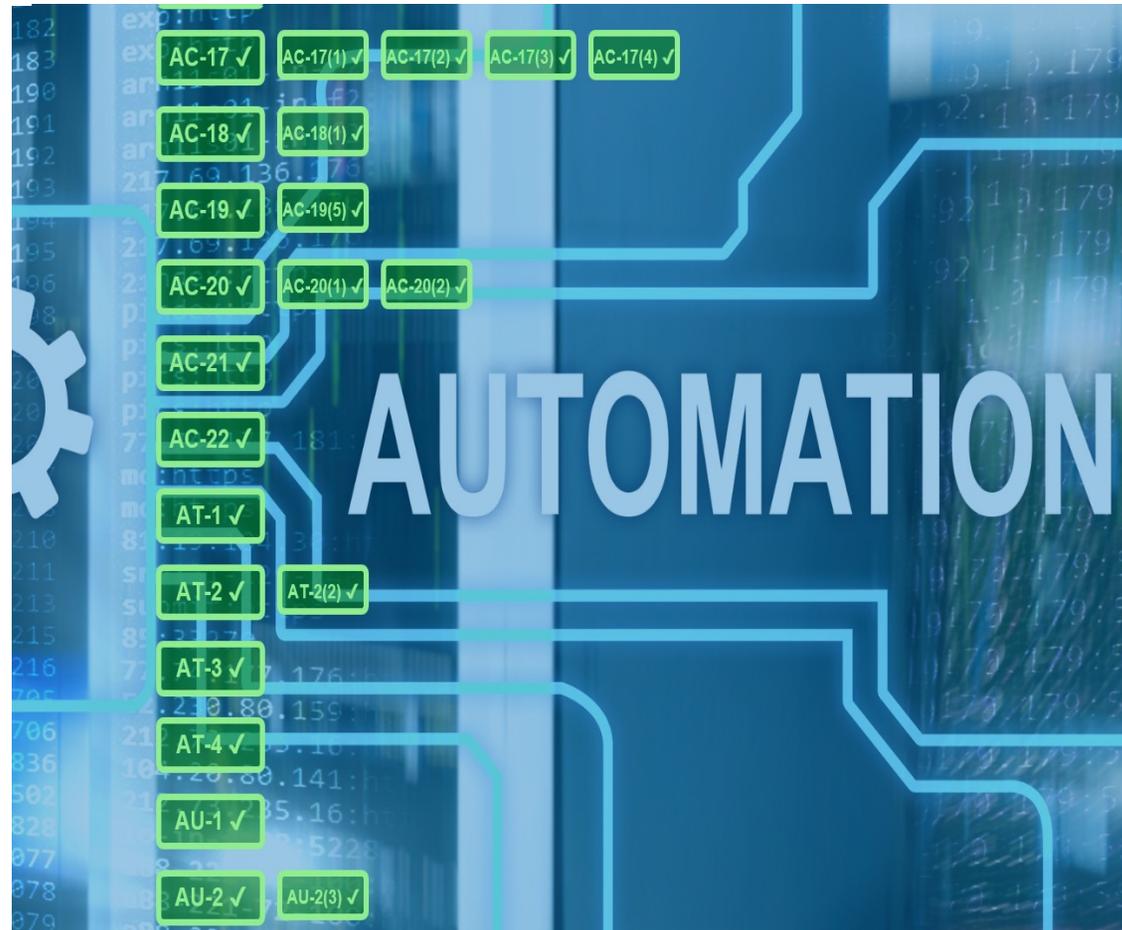
Continuous Assessment Approach

- **Assessment Plan:** What should be tested/inspected, how, and with which cadence is desired
- **Assessment Results:** Time-slice of results



OSCAL POA&M Model





What can you
do with the
OSCAL models?

OSCAL Models >>> OSCAL Content >>> OSCAL Tools

```

catalog [1]: {
  uuid [1]: uuid,
  metadata [1]: { - },
  params [0 or 1]: [ - ],
  controls [0 or 1]: [ - ],
  groups [0 or 1]: [ - ],
  back-matter [0 or 1]: { - },
}
profile [1]: {
  uuid [1]: uuid,
  metadata [1]: { - },
  imports [1]: [ - ],
  merge [0 or 1]: { - },
  modify [0 or 1]: { - },
  back-matter [0 or 1]: { - },
}
component-definition [1]: {
  uuid [1]: uuid,
  metadata [1]: { - },
  imports-component-definitions [0 or 1]: [ - ],
  components [0 or 1]: [ - ],
  capabilities [0 or 1]: [ - ],
  back-matter [0 or 1]: { - },
}
system-security-plan [1]: {
  uuid [1]: uuid,
  metadata [1]: { - },
  profile [1]: { - },
  oscal-changes [0 or 1]: { - },
  control-implementation [1]: { - },
  back-matter [0 or 1]: { - },
}
assessment-plan [1]: {
  uuid [1]: uuid,
  metadata [1]: { - },
  import-ssp [1]: { - },
  local-definitions [0 or 1]: { - },
  terms-and-conditions [0 or 1]: { - },
  reviewed-controls [1]: { - },
  assessment-subjects [0 or 1]: [ - ],
  assessment-assets [0 or 1]: { - },
  tasks [0 or 1]: [ - ],
  back-matter [0 or 1]: { - },
}
assessment-results [1]: {
  uuid [1]: uuid,
  metadata [1]: { - },
  import-ap [1]: { - },
  local-definitions [0 or 1]: { - },
  results [1]: [ - ],
  back-matter [0 or 1]: { - },
}
plan-of-action-and-milestones [1]: {
  uuid [1]: uuid,
  metadata [1]: { - },
  import-ssp [0 or 1]: { - },
  system-id [0 or 1]: { - },
  local-definitions [0 or 1]: { - },
  observations [0 or 1]: [ - ],
  risks [0 or 1]: [ - ],
  poam-items [1]: [ - ],
  back-matter [0 or 1]: { - },
}
  
```

OSCAL Models

<https://github.com/usnistgov/OSCAL>

usnistgov / oscal-content Public

Code Issues 22 Pull requests 2

master oscal-content / nist.gov / SP800-53 / rev5 / xml /

OSCAL Content Generation

OSCAL Content in Action

NIST_SP-800-53_rev5_HIGH-baseline-resolved-profile...

NIST_SP-800-53_rev5_LOW-baseline-resolved-profile...

NIST_SP-800-53_rev5_MODERATE-baseline-resolved-...

NIST_SP-800-53_rev5_PRIVACY-baseline-resolved-pr...

NIST_SP-800-53_rev5_PRIVACY-baseline_profile.xml

NIST_SP-800-53_rev5_catalog.xml

<https://github.com/usnistgov/oscal-content>

Name	Provider/Developer	Description	Type
Compliance trestle	IBM	A python SDK and command line tool which manipulates OSCAL structures and supports transformation of data into OSCAL.	open source
OSCAL Java Library	NIST OSCAL Project	A Java-based programming API for reading and writing content conformant to the OSCAL XML, JSON, and YAML based models.	open source
OSCAL React Component Library	Easy Dynamics	A library of reusable React components and an example user interface application that provides a direct UI into OSCAL.	open source
XSLT Tooling	NIST OSCAL Project	A variety of Extensible Stylesheet Transformations (XSLT) Sheets (CSS), and related utilities for authoring, converting, and publishing OSCAL content in various forms.	open source
XML Jelly Sandwich	Wendell Piez (NIST)	Interactive XSLT in the browser includes OSCAL demonstrations .	open source
Xacta 360	C2 Labs	Xacta 360 is a cyber risk management and compliance platform that provides analysis and support for system security plans (SSPs) in OSCAL format. Future OSCAL capabilities are forthcoming as the platform evolves.	community edition
Atlassian: Continuous Compliance Automation	C2 Labs	Atlassian: Continuous Compliance Automation runs in any environment and supports the development of OSCAL v1.0 content for Catalogs, Profiles, System Security Plans and Components. Additional detail can be found in this blog post: Atlassian Delivers Free Tools to Create OSCAL Content .	community edition

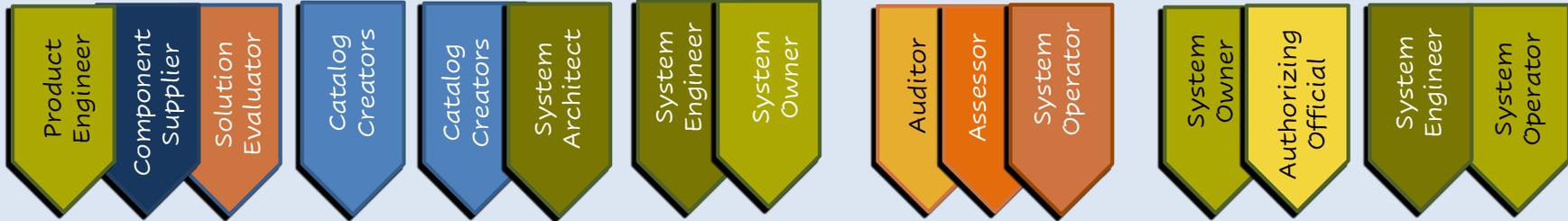
<https://github.com/usnistgov/oscal-tools>

OSCAL Editorial Tools

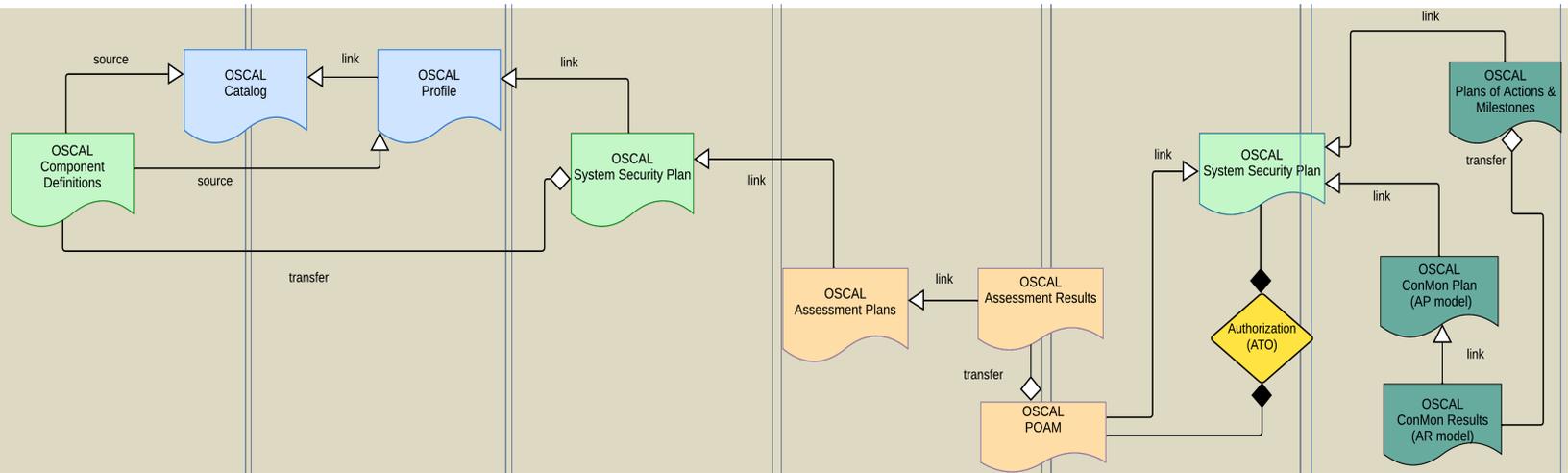
OSCAL GRC Tools

Who can benefit & How

Actors



Risk Management & OSCAL content



RMF steps: PREPARE

CATEGORIZE

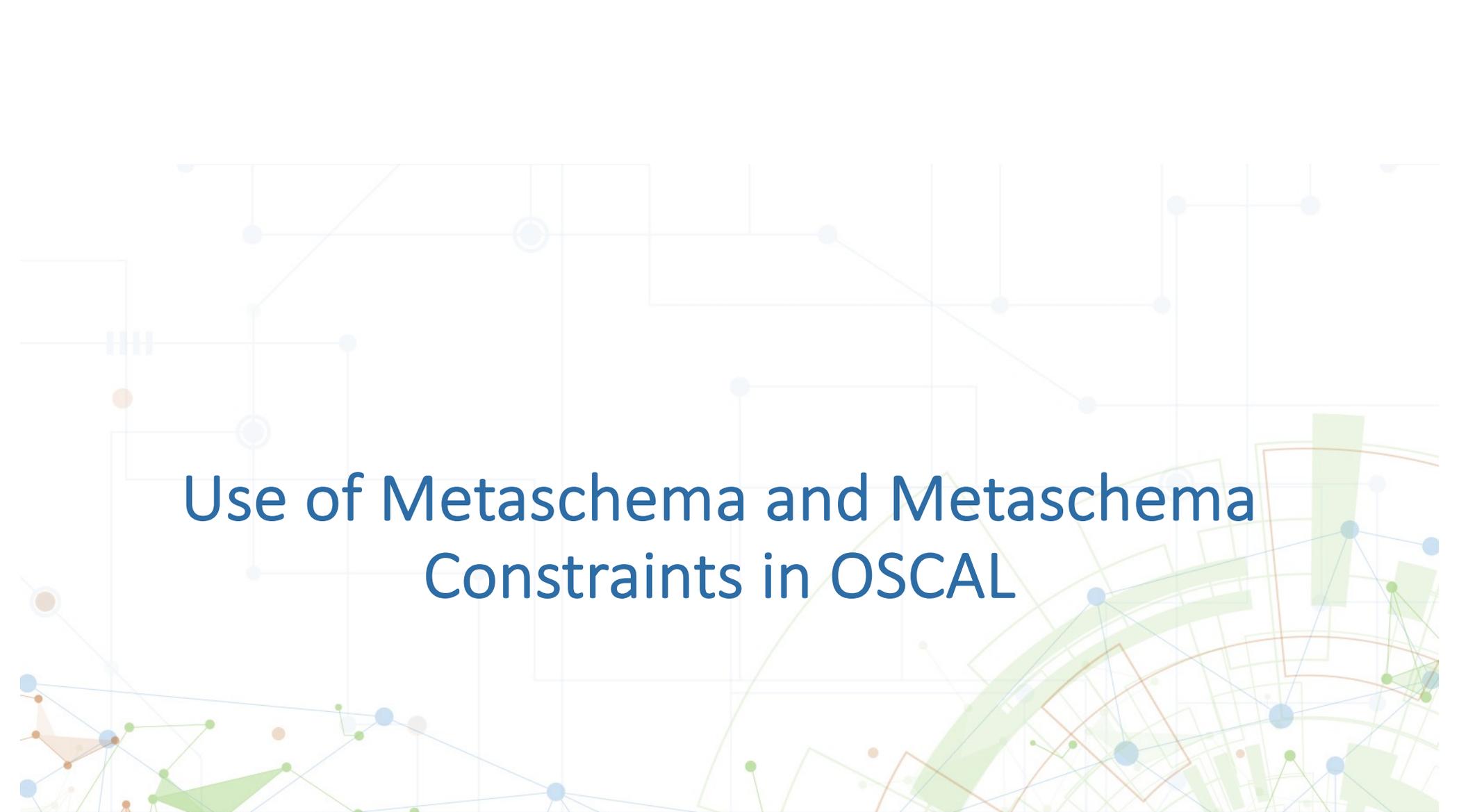
SELECT

IMPLEMENT

ASSESS

AUTHORIZE

CON-MON

The background features a complex network of nodes and lines in various colors (blue, green, orange, grey) overlaid on a light grid. The nodes are represented by small circles, and the lines are thin, connecting the nodes in a web-like structure. Some nodes are highlighted with larger, semi-transparent shapes in shades of green and blue.

Use of Metaschema and Metaschema Constraints in OSCAL

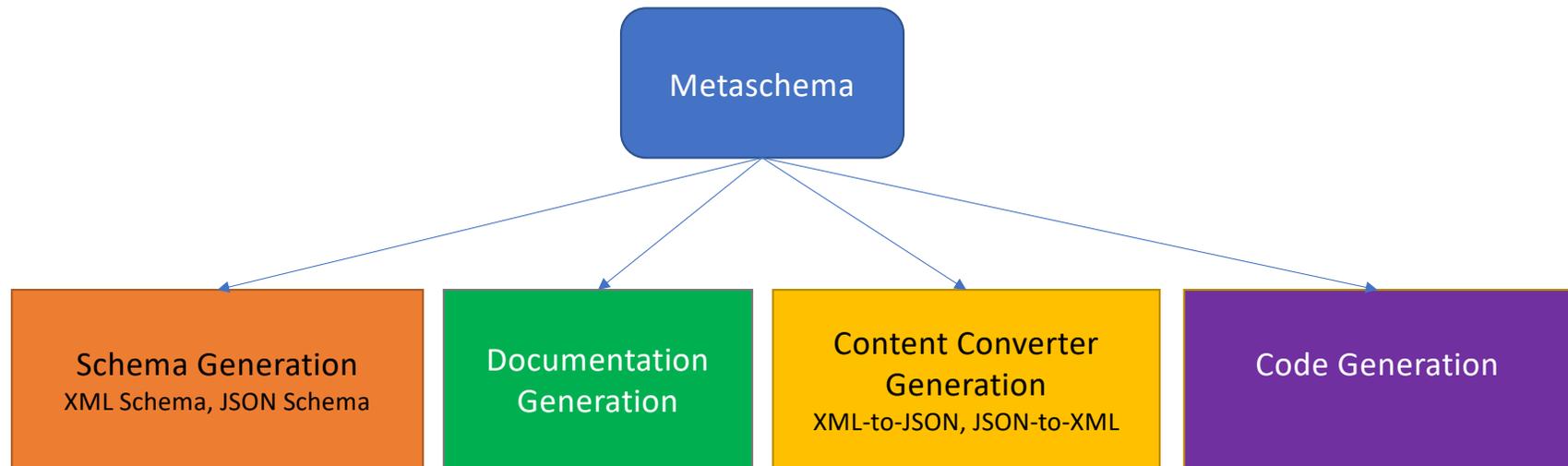
Metaschema provides a format-agnostic foundation for managing data formats and supporting tooling



- OSCAL uses Metaschema to define the logical structure of OSCAL data.
- Metaschema is format agnostic. We can support XML, JSON, and YAML with no extra effort.
- XML and JSON Schema are derived from Metaschema definitions to support well-formedness checking for XML, JSON, and YAML formats.
- Model documentation can be auto-generated.

Metaschema provides a format-agnostic foundation for managing the OSCAL models.

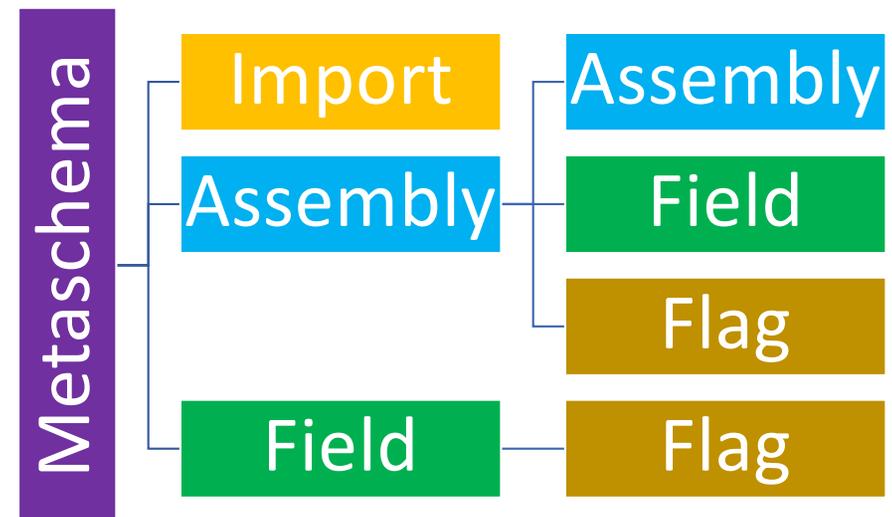
Use of Metaschema provides model agility



A single modeling approach supporting multiple formats and related productions.

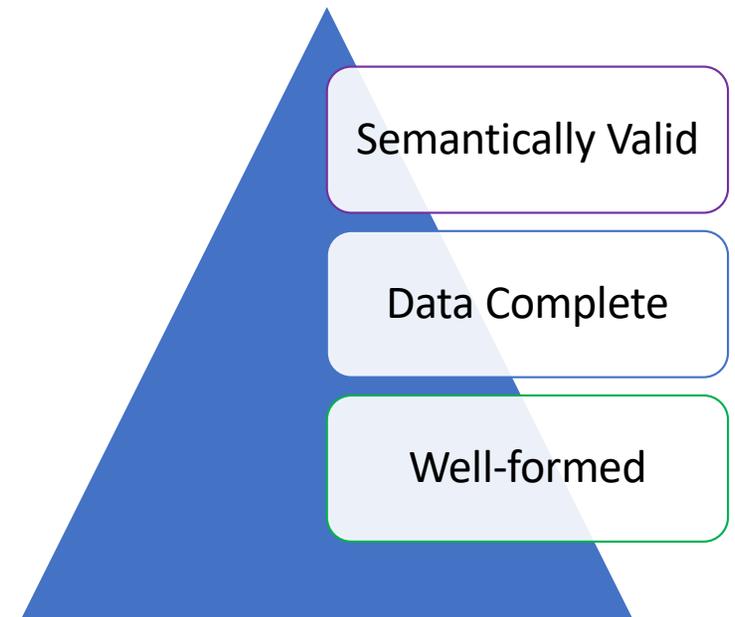
Metaschema structure

- Metaschema is compositional.
- **Imports** allow constructs to be shared between Metaschemas.
- **Assembly:** Represents a data object.
 - Contains child assemblies and fields at various cardinalities
 - May contain flags
- **Field:** Represents a valued data point
 - May contain flags
- **Flag:** Qualifies or supports a data object



The path to richer Semantics in OSCAL

- Metaschema supports format-agnostic constraints.
 - Supports co-constraints (i.e., if X then Y)
 - Richer rules than schema validation alone
- Constraints are written in Metapath, an XPath-like grammar.
- Can measure data completeness for a given use case
- Will allow different dialects to be defined for OSCAL (i.e., FedRAMP)



Open-Source Tools and Libraries

<https://pages.nist.gov/OSCAL/tools/#open-source-tools-and-libraries>

Name	Provider/Developer	Description	Type
Compliance trestle ↗	IBM	A python SDK and command line tool which manipulates OSCAL structures and supports transformation of data into OSCAL.	open source
OSCAL Java Library ↗	NIST OSCAL Project	A Java-based programming API for reading and writing content conformant to the OSCAL XML, JSON, and YAML based models.	open source
OSCAL React Component Library ↗	Easy Dynamics	A library of reusable React components and an example user interface application ↗ that provides a direct UI into OSCAL.	open source
OSCAL REST API ↗	Easy Dynamics	An initial OpenAPI definition of an OSCAL REST API that describes how systems might manipulate catalogs, profiles, components, and SSPs.	open source
XSLT Tooling ↗	NIST OSCAL Project	A variety of Extensible Stylesheet Language (XSL) Transformations (XSLT), Cascading Style Sheets (CSS), and related utilities for authoring, converting, and publishing OSCAL content in various forms.	open source
XML Jelly Sandwich ↗	Wendell Piez (NIST)	Interactive XSLT in the browser includes OSCAL demonstrations ↗ .	open source
Xacta 360 ↗	Telos	Xacta 360 is a cyber risk management and compliance analytics platform that enables users to create and submit FedRAMP system security plans (SSPs) in OSCAL format. Future OSCAL capabilities are forthcoming as the standard evolves.	license ↗
Atlasity: Continuous Compliance Automation ↗	C2 Labs	Atlasity CE (release 2.0) runs in any environment and supports the development of OSCAL v1.0 content for Catalogs, Profiles, System Security Plans and Components. Additional detail can be found in this blog post: Atlasity Delivers Free Tools to Create OSCAL Content ↗ .	community edition
control_freak ↗	Risk Redux	This tool seeks to provide folks with a searchable and easy-to-navigate reference for NIST SP 800-53 Revision 5. It is an open-source application from the Risk Redux project ↗ , built using parsed content directly from the OSCAL repositories.	open-source

Few of the OSCAL Adopters

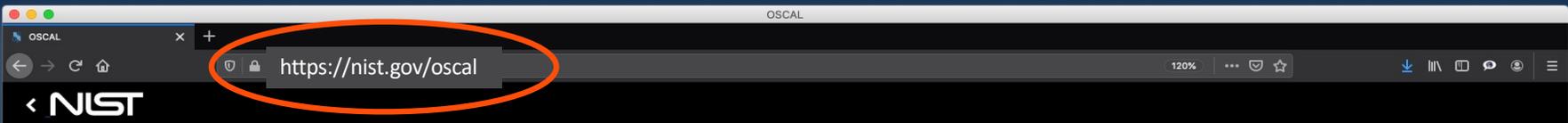


- 2021 presenters**
- FedRAMP
 - Noblis
 - HHS CMS
 - National Renewable Energy Lab
 - GovReady
 - C2 Labs
 - cFocus Software
 - Shujinko
 - Robers Bosch (EU | Germany)
 - Telos
 - KPMG
 - IBM Research

- 2022 new presenters**
- AWS
 - CSAM
 - Easy Dymanics
 - Volant Associates LLC
 - Secureframe
 - Red Hat
 - Nirmata
 - SunStone Secure

- 2021-2022 other adopters**
- Booz Allen Hamilton
 - eMASS
 - Microsoft
 - Coalfire
 - Kratos
 - Salesforce
 - Oracle





OSCAL: the Open Security Controls Assessment Language

[Get involved](#) | [Contact Us](#) | [Github](#)

- [Learn More](#)
- [Tutorials](#)
- [Tools](#)
- [Documentation](#)
- [Downloads](#)
- [Contribute](#)
- [Contact Us](#)

Automated Control-Based Assessment

Supporting Control-Based Risk Management with Standardized Formats

[Learn More](#)



AUTOMATION

Providing control-related information in machine-readable formats.

NIST, in collaboration with industry, is developing the Open Security Controls Assessment Language (OSCAL). OSCAL is a set of formats expressed in XML, JSON, and YAML. These formats provide machine-readable representations of control catalogs, control baselines, system security plans, and assessment plans and results.

Publicly Available Resources



Documentation:

Catalog, Profile, Component, SSP, SAP, SAR, POA&M:

<https://pages.nist.gov/OSCAL/documentation/>



Example:

Generic examples:

<https://github.com/usnistgov/oscal-content/tree/master/examples>

NIST SP 800-53 R4 and Rev5 catalog and baselines (XML & JSON):

<https://github.com/usnistgov/oscal-content/tree/master/nist.gov/SP800-53>



FedRAMP

FedRAMP Automation:

Repository (FedRAMP catalog and baselines (XML & JSON) included) :

<https://github.com/GSA/fedramp-automation>

<https://www.fedramp.gov/using-the-fedramp-oscal-resources-and-templates/>



Tools

OSCAL Java Library: <https://github.com/usnistgov/liboscal-java>

XSLT Tooling: <https://github.com/usnistgov/oscal-tools/tree/master/xslt>

OSCAL Kit: <https://github.com/docker/oscalkit>

OSCAL GUI: <https://github.com/brianrufigsa/OSCAL-GUI>

OMB'S OPAL: OSCAL Policy Administration Library (OPAL): <https://github.com/EOP-OMB/opal>

Please visit Community's:
OSCAL Club/awesome-oscal:

<https://github.com/oscal-club/awesome-oscal>

Questions?



Contact us at: oscal@nist.gov

Chat with us on Gitter: <https://gitter.im/usnistgov-OSCAL/Lobby>

Collaborate with us on GitHub: <https://github.com/usnistgov/OSCAL>

Join our COI meetings: <https://pages.nist.gov/OSCAL/contribute/#community-meetings>

Thank you!