

Mapping: FedRAMP Rules ? OSCAL

DRAFT - PLEASE PROVIDE INPUT

Updated: September 8, 2026 [Latest Work Found Here](#)

This document maps every field defined in the `FRR` portion of `fedramp-consolidated-rules.schema.json` to its corresponding location in the OSCAL catalog produced by `src/frr2oscal.py`. Fields with no current mapping are marked **NOT MAPPED** and noted with a brief description of the data they contain.

The catalog metadata also includes fixed roles, parties, and responsible-parties that are not derived from the source JSON; these are described in §9.

1. Schema Hierarchy Overview

FRR	(object, keyed by <code>frr_document_key</code>)
└─ {FRR-key}	e.g. "AFC", "AGU"
└─ info	<code>frr_document_info</code>
└─ data	<code>data_container_frr</code>
└─ all	<code>frr_requirements_map</code> ← processed
└─ 20x	<code>frr_requirements_map</code> ← processed
└─ rev5	<code>frr_requirements_map</code> ← processed
└─ {subset-key}	<code>frr_requirements_subset_group</code>
└─ {rule-id}	<code>frr_requirement</code>
└─ <code>varies_by_class.{a b c d}</code>	<code>frr_requirement_level</code>

All three data scopes (`all`, `20x`, `rev5`) are now converted. Rules from each scope carry a `path` prop (FedRAMP namespace) identifying which scope they belong to. Child group IDs include a scope qualifier for `20x` and `rev5` to prevent collisions with `all` groups whose subset keys overlap (e.g. VDR/TFR appears in all three scopes).

2. FRR Document ? OSCAL Top-Level Group

Each top-level key in `FRR` (pattern `^[A-Z]{3}$`) becomes a top-level `group` in the catalog.

2.1 FRR.{key} ? group

OSCAL field	Value
group.id	"FRR-{key}"
group.title	info.name
group.props[name=label].value	"FRR-{key}"
group.parts[name=overview].prose	info.purpose
group.parts[name=overview].title	"Purpose" (set via dict workaround; see MISSING_FROM_OSCAL_LIBRARY.md §1)

2.2 FRR.{key}.info fields (frr_document_info)

Schema field	Required	Type	OSCAL mapping
name	✓	string	→ group.title
purpose	✓	string	→ group.parts[name=overview].prose
short_name	✓	string <code>^[A-Z]{3}\$</code>	→ group.props[name=label].value
web_name	✓	string	→ group.props[name=web_name, ns=FRR_NS].value (FedRAMP extension)
status	✓	"stable" "placeholder" "empty"	→ group.props[name=status, ns=FRR_NS].value (FedRAMP extension)
tag	—	string	→ group.props[name=tag, ns=FRR_NS].value (FedRAMP extension)
effective	✓ (or 20x + rev5)	effective_entry	NOT MAPPED — effective date and status for the ruleset
subsets	—	frr_info_subsets	Mapped as child controls — subset metadata (names, descriptions, applicability); see §2.3
flows	—	array	NOT MAPPED — process-flow relationships
20x	✓ (when no top-level effective)	frr_document_info_certification	NOT MAPPED — 20x-specific effective dates and subset overrides

Schema field	Required	Type	OSCAL mapping
rev5	✓ (when no top-level effective)	frr_document_info_certification	NOT MAPPED — Rev5-specific effective dates and subset overrides

2.3 FRR.{key}.info.subsets (frr_info_subsets)

Subset metadata describes each named group of rules within a ruleset. It is present in the `info` block but is not present in the current data for any ruleset. When present, each entry becomes a child `control` of the parent ruleset group (id = `FRR-{key}-{subset-key}`).

Schema field	Required	Type	OSCAL mapping
{subset-key}.name	✓	string	→ <code>control.title</code> (prefixed with "SUBSET: ")
{subset-key}.description	✓	string	→ <code>control.parts[name=statement].prose</code>
{subset-key}.applicability.types	✓	["20x" "Rev5"]	→ one <code>control.props[name=applicability-type, ns=FRR_NS]</code> per item
{subset-key}.applicability.paths	✓	["Program" "Agency"]	→ one <code>control.props[name=applicability-path, ns=FRR_NS]</code> per item
{subset-key}.applicability.classes	✓	["A" ... "D"]	→ one <code>control.props[name=applicability-class, ns=FRR_NS]</code> per item
{subset-key}.applicability.affects	✓	affected_party[]	→ one <code>control.props[name=affects, ns=FRR_NS]</code> per item

3. FRR.{key}.data ? Data Scope Selection

All three scopes are processed. Child group and control IDs reflect the scope they originate from.

Schema key	Type	OSCAL mapping
data.all	frr_requirements_map	→ processed; child groups use <code>FRR-{key}-{subset}</code> ID
data.20x	frr_requirements_map	→ processed; child groups use <code>FRR-{key}-{subset}-20x</code> ID

Schema key	Type	OSCAL mapping
data.rev5	frr_requirements_map	→ processed; child groups use FRR- {key}-{subset}-rev5 ID

4. FRR.{key}.data.{scope}.{subset} ? OSCAL Child Group

Each subset key within a scope becomes a child `group` nested under the parent ruleset group.

OSCAL field	Value
group.id	"FRR-{key}-{subset}" (scope=all) or "FRR-{key}-{subset}-{scope}" (scope=20x/rev5)
group.title	Same as group.id for scope=all; "FRR-{key}-{subset} 20X Path" or "FRR-{key}-{subset} Rev 5 Path" for scoped groups
group.parts[name=overview]	Omitted (no info block present in current data)

“The `frr_requirements_subset_group` schema allows any `frr_requirement_id` as a property key alongside an optional `info` entry. When `info` is present, its `title` and `purpose` fields map to `group.title` and `group.parts[name=overview].prose` respectively.

5. FRR.{key}.data.{scope}.{subset}.rule-id ? OSCAL Control

Each rule entry (`frr_requirement`) maps to one OSCAL `control`. The schema enforces a mutual exclusion: a rule has either `statement` + `force` (simple rule) or `varies_by_class` (class-conditional rule), never both.

5.1 Simple Rule (has `statement` and `force`, no `varies_by_class`)

Schema field	Required	Type	OSCAL mapping
(key)	—	frr_requirement_id	→ control.id, props[name=label].value
name	✓	string	→ control.title
statement	✓	string	→ parts[name=statement, id={id}_smt].prose

Schema field	Required	Type	OSCAL mapping
<code>force</code>	✓	<code>force_enum</code>	→ <code>props[name=force, ns=FRR_NS].value</code>
<i>(scope)</i>	—	—	→ <code>props[name=path, ns=FRR_NS].value = "all" "20x" "rev5"</code>
<code>affects</code>	✓	<code>affected_party[]</code>	→ one <code>props[name=affects, ns=FRR_NS]</code> per item
<code>danger</code>	—	string	→ <code>parts[name=guidance, title=Danger].prose</code> (title set via workaround; see MISSING_FROM_OSCAL_LIBRARY.md §1)
<code>note</code>	—	string	→ <code>parts[name=guidance, title=Notes].prose</code>
<code>notes</code>	—	string[] (min 2)	→ <code>parts[name=guidance, title=Notes].prose</code> (items joined with blank lines)
<code>related</code>	—	string[]	→ one <code>links[rel=related, href=#{id}]</code> per item
<code>artifacts.all</code>	—	string[]	→ <code>parts[name=assessment-method, props=[method=EXAMINE, path=all]].parts[name=assessment-objects].prose</code> (one per item; see MISSING_FROM_OSCAL_LIBRARY.md §3)
<code>artifacts.20x</code>	—	string[]	→ <code>parts[name=assessment-method, props=[method=EXAMINE, path=20x]].parts[name=assessment-objects].prose</code>
<code>artifacts.rev5</code>	—	string[]	→ <code>parts[name=assessment-method, props=[method=EXAMINE, path=rev5]].parts[name=assessment-objects].prose</code>
<code>corrective_actions</code>	—	string[]	→ <code>parts[name=remediation, ns=FRR_NS].prose</code> (items joined with blank lines)
<code>examples</code>	—	<code>object[] (id, key_tests, examples)</code>	→ <code>parts[name=guidance, class=examples, title=Examples].prose</code>
<code>schema</code>	—	<code>rule_schema (name + url)</code>	→ <code>parts[name=guidance, class=schema, title=Schema].prose</code>

Schema field	Required	Type	OSCAL mapping
following_information	—	string[]	→ parts[name=guidance, class=following_information, title=Following Information].prose
updated	✓	updated_list	→ one props[name=updated, ns=FRR_NS, value=date, remarks=comment] per entry
following_information_bullets	—	string[]	→ parts[name=statement, id={id}_smt].parts[name=item].prose (each item prefixed with -; appended after statement)
reference	—	string	→ links[rel=reference].text (used as link text when reference_url is also present)
reference_url	—	URI	→ links[rel=reference, href={url}]; .text set to reference value when present
effective_date	—	effective_dates	→ props[name=effective_date, ns=FRR_NS].value (JSON-serialised when value is an object)
timeframe_type	—	enum (bizdays days hours weeks months years)	→ props[name=timeframe_type, ns=FRR_NS].value
timeframe_num	—	positive number	→ props[name=timeframe_num, ns=FRR_NS].value (number converted to string)
notification	—	object[] (party, method, target, name, url)	NOT MAPPED — structured notification requirements
controls	—	control_id[]	→ one links[rel=related, href=#{NIST_RESOURCE_UUID}, resource-fragment={id}, text="NIST SP 800-53 Rev 5 {ID}"] per item (see §10)
terms	—	string[]	NOT MAPPED — FRD term references

5.2 Varies-by-Class Rule (has varies_by_class, no statement or force)

The rule becomes a parent `control` with a fixed statement, and each class variant becomes a child `control` nested in the parent's `controls` list (see `MISSING_FROM_OSCAL_LIBRARY.md` §4 for the library workaround).

Parent control:

Schema field	Required	Type	OSCAL mapping
<i>(key)</i>	—	<code>frr_requirement_id</code>	→ <code>control.id</code> , <code>props[name=label].value</code>
<code>name</code>	✓	string	→ <code>control.title</code>
<i>(hardcoded)</i>	—	—	→ <code>parts[name=statement].prose</code> = "Varies by Class"
<i>(scope)</i>	—	—	→ <code>props[name=path,</code> <code>ns=FRR_NS].value</code> = "all" "20x" "rev5"
<code>related</code>	—	string[]	→ one <code>links[rel=related,</code> <code>href=#{id}]</code> per item
<code>affects</code>	✓	<code>affected_party[]</code>	→ one <code>props[name=affects,</code> <code>ns=FRR_NS]</code> per item
<code>note</code>	—	string	→ <code>parts[name=guidance,</code> <code>title=Notes].prose</code>
<code>notes</code>	—	string[]	→ <code>parts[name=guidance,</code> <code>title=Notes].prose</code> (items joined with blank lines)
<code>corrective_actions</code>	—	string[]	→ <code>parts[name=remediation,</code> <code>ns=FRR_NS].prose</code>
<code>examples</code>	—	object[]	→ <code>parts[name=guidance,</code> <code>class=examples,</code> <code>title=Examples].prose</code>
<code>schema</code>	—	<code>rule_schema</code>	→ <code>parts[name=guidance,</code> <code>class=schema,</code> <code>title=Schema].prose</code>
<code>following_information</code>	—	string[]	→ <code>parts[name=guidance,</code> <code>class=following_information</code> <code>, title=Following</code> <code>Information].prose</code>
<code>updated</code>	✓	<code>updated_list</code>	→ one <code>props[name=updated,</code> <code>ns=FRR_NS, value=date,</code> <code>remarks=comment]</code> per entry
<code>following_information_bullets</code>	—	string[]	→ <code>parts[name=statement,</code> <code>id={id}_smt].parts[name=ite</code> <code>m].prose</code> (each item prefixed with <code>-</code>)

Schema field	Required	Type	OSCAL mapping
reference	—	string	→ <code>links[rel=reference].text</code> (used as link text when <code>reference_url</code> is also present)
reference_url	—	URI	→ <code>links[rel=reference, href={url}].text</code> set to <code>reference</code> value when present
effective_date	—	effective_dates	→ <code>props[name=effective_date, ns=FRR_NS].value</code> (JSON-serialised when value is an object)
timeframe_type	—	enum	→ <code>props[name=timeframe_type, ns=FRR_NS].value</code>
timeframe_num	—	positive number	→ <code>props[name=timeframe_num, ns=FRR_NS].value</code> (number converted to string)
notification	—	object[]	NOT MAPPED
controls	—	control_id[]	→ one <code>links[rel=related, href=#{NIST_RESOURCE_UUID}, resource-fragment={id}, text="NIST SP 800-53 Rev 5 {ID}"]</code> per item (see §10)
terms	—	string[]	NOT MAPPED

6. `varies_by_class.{a|b|c|d}` ? OSCAL Child Control (`frr_requirement_level`)

Each class key (`a`, `b`, `c`, `d`) becomes a child `control` appended directly to the parent control's `controls` list.

Schema field	Required	Type	OSCAL mapping
(key, uppercased)	—	"a" ... "d"	→ <code>control.id = "{parent-id}-{key}"</code> , <code>props[name=label].value = "Class {KEY}"</code>
(label as title)	—	—	→ <code>control.title = "Class {KEY}"</code> (class entries have no <code>name</code> field)

Schema field	Required	Type	OSCAL mapping
(scope)	—	—	→ <code>props[name=path, ns=FRR_NS].value</code> (inherited from parent rule's scope)
statement	✓	string	→ <code>parts[name=statement, id={id}_smt].prose</code>
force	✓	force_enum	→ <code>props[name=force, ns=FRR_NS].value</code>
note	—	string	→ <code>parts[name=guidance, title=Notes].prose</code>
notes	—	string[] (min 2)	→ <code>parts[name=guidance, title=Notes].prose</code> (items joined with blank lines)
artifacts.all	—	string[]	→ <code>parts[name=assessment-method, props=[method=EXAMINE, path=all]].parts[name=assessment-objects].prose</code>
artifacts.20x	—	string[]	→ <code>parts[name=assessment-method, props=[method=EXAMINE, path=20x]].parts[name=assessment-objects].prose</code>
artifacts.rev5	—	string[]	→ <code>parts[name=assessment-method, props=[method=EXAMINE, path=rev5]].parts[name=assessment-objects].prose</code>
following_information	—	string[]	→ <code>parts[name=guidance, class=following_information, title=Following Information].prose</code>
following_information_bullets	—	string[]	→ <code>parts[name=statement, id={id}_smt].parts[name=item].prose</code> (each item prefixed with <code>-</code>)
rev5_controls_list	—	object (family → <code>rev5_control_id[]</code>)	→ used for Rev5 profile control selection (not written to the catalog); control IDs are converted to OSCAL format and added to the appropriate Rev5 class profile's include list
effective_date	—	effective_dates	→ <code>props[name=effective_date, ns=FRR_NS].value</code> (JSON-serialised when value is an object)

Schema field	Required	Type	OSCAL mapping
timeframe_type	—	enum	→ props[name=timeframe_type, ns=FRR_NS].value
timeframe_num	—	positive number	→ props[name=timeframe_num, ns=FRR_NS].value (number converted to string)
pain_timeframes	—	object (PAIN level 1-5 → timeframe)	NOT MAPPED — PAIN score-based response timeframe table

7. Supporting Types Summary

Type	Used in	Description
force_enum	rule, class variant	MUST MUST NOT SHOULD SHOULD NOT MAY
affected_party	rule, subset applicability	Advisors Agencies Assessors FedRAMP Providers Everyone
certification_type	subset applicability	20x Rev5
certification_path	subset applicability	Program Agency
class_name	subset applicability	A B C D
timeframe_type	rule, class variant	bizdays days hours weeks months years
updated_list	rule	Array of {date, comment} changelog entries
effective_entry	document info	{is, current_status, date, comments, warnings, signup_url}
effective_dates	rule, class variant	{obtain, maintain, optional_adoption, grace} dates
notification	rule	{party, method, target, name, url} — who to notify, how, and where
rule_schema	rule	{name, url} — machine-readable schema governing artifacts
pain_timeframes	class variant	PAIN severity 1-5 → response timeframe table
rev5_controls_list	class variant	NIST SP 800-53 Rev5 family → control ID list
artifact_list	artifacts	string[] — plain-text artifact descriptions

Type	Used in	Description
control_id	rule	<code>^[a-z]{2}-\d+(\.\d+)?\$</code> — NIST SP 800-53 control ID

8. Summary of Unaddressed Fields

The following fields appear in the source data but are not yet mapped to OSCAL output. They are recorded in `data/unhandled.json` at runtime.

Field	Appears in	Description
notification	rule	Structured notification requirements (party, method, target, URL)
terms	rule	References to defined terms in the FRD
pain_timeframes	class variant	PAIN severity 1-5 → response timeframe lookup table

9. Catalog Metadata — Roles, Parties, and Responsible-Parties

The following entries are written to the catalog `metadata` section by `_add_catalog_contacts()` regardless of source-JSON content. They are not derived from the FRR data.

9.1 Roles

Role id	title
fedramp	FedRAMP
system-owner	System Owner
assessor	Assessor
agency	Agency

9.2 Parties

name	type	uuid constant	Extra fields
FedRAMP PMO	organization	<code>_PARTY_UUID_FEDRAMP</code>	<code>email-addresses: [info@fedramp.gov], links[rel=website, href=https://www.fedramp.gov]</code>

name	type	uuid constant	Extra fields
Cloud Service Provider	organization	<code>_PARTY_UUID_CSP</code>	—
Assessing Organization	organization	<code>_PARTY_UUID_AO</code>	—
Federal Agency	organization	<code>_PARTY_UUID_AGENCY</code>	—

9.3 Responsible-Parties

role-id	party-uuids
<code>fedramp</code>	<code>[_PARTY_UUID_FEDRAMP]</code>
<code>system-owner</code>	<code>[_PARTY_UUID_CSP]</code>
<code>assessor</code>	<code>[_PARTY_UUID_AO]</code>
<code>agency</code>	<code>[_PARTY_UUID_AGENCY]</code>

10. Catalog Back-Matter — NIST SP 800-53 Rev 5 Resource

A single back-matter resource is added to the catalog by `build_catalog()` so that `controls` citations on rules can reference it via UUID fragment. The resource UUID is the stable module-level constant `_NIST_800_53_REV5_RESOURCE_UUID`.

Field	Value
<code>uuid</code>	<code>_NIST_800_53_REV5_RESOURCE_UUID</code> (<code>ffffffff-0000-4000-a000-000000000001</code>)
<code>title</code>	"NIST SP 800-53 Rev 5"
<code>rlinks[0].href</code>	<code>NIST_800_53_REV5_URL</code> — OSCAL JSON on GitHub raw
<code>rlinks[1].href</code>	<code>NIST_800_53_REV5_DOI_URL</code> — canonical DOI (<code>https://doi.org/10.6028/NIST.SP.800-53r5</code>)

Each NIST control ID listed in a rule's `controls` array produces a link on the OSCAL control:

```
links[
  rel          = "related",
  href         = "#{NIST_RESOURCE_UUID}",
  resource-fragment = {control-id},
  text         = "NIST SP 800-53 Rev 5 {CONTROL-ID}", ← id uppercased
]
```

The `resource-fragment` field identifies the specific control within the referenced catalog without requiring a separate per-control back-matter entry. The `text` field provides a

human-readable label with the control ID in uppercase.

Revision #5

Created 2026-08-20 19:34:44 UTC by Brian Ruf

Updated 2026-09-22 02:21:47 UTC by Brian Ruf