



OSCAL POCKET GUIDE

A Mobile-First OSCAL Consumer and Producer
for the NIST Compliance Ecosystem

Tevin Harris · euCann LLC · App Version 1.5.1

Six NIST frameworks. Offline-first. OSCAL in, OSCAL out.

01

THE PROBLEM

Compliance Is Hard. The Tools Don't Help.

The Practitioner's Reality

- 1,193+ controls navigated via PDFs and spreadsheets
- Baseline selection done manually, disconnected from authoritative data
- SSP creation means Word templates + copy-paste from PDFs for weeks
- Cross-framework work requires juggling multiple documents simultaneously
- None of this works offline, on mobile, or in air-gapped environments

Why OSCAL Should Solve This

- OSCAL catalogs carry exactly the data practitioners need
- NIST published high-quality JSON for all major frameworks
- **But practitioners have never interacted with OSCAL directly — they need tools, not file formats**

"What if the OSCAL catalog was the database powering the tool practitioners actually use every day?"

02

THE SOLUTION

What Is OSCAL Pocket Guide?

A Cross-Platform Flutter App for the NIST Ecosystem

| Attribute | Detail |
|--------------|--|
| Framework | Flutter 3.35 / Dart 3.7 |
| Platforms | iOS, Android, Web |
| Data Format | OSCAL JSON (native — not a conversion layer) |
| Architecture | Offline-first, bundled catalogs + SQLite/SQLCipher |
| Auth & Sync | Supabase (Google, Apple, email) |
| Status | v1.5.1 — iOS App Store & Google Play |

Core thesis: OSCAL catalogs are not just a publishing format — they are a runtime data model. The app parses official NIST OSCAL JSON at startup and uses it as the live database for every feature.

Six NIST Frameworks, One App

All Powered by Official OSCAL JSON

| # | Framework | OSCAL Source | Contents |
|---|-------------------|--|---|
| 1 | SP 800-53 Rev 5 | NIST_SP-800-53_rev5_catalog.json | 1,000+ controls, 20 families, 4 baseline profiles |
| 2 | NIST CSF 2.0 | NIST_CSF_v2.0_catalog.json | 6 Functions, 22 Categories, 106 Subcategories |
| 3 | SP 800-171 Rev 3 | NIST_SP800-171_rev3_catalog.json | CUI protection, 17 families, OSCAL 1.2.0 |
| 4 | SP 800-218 (SSDF) | NIST_SP800-218_ver1_catalog.json | 4 groups, 19 practices, 42+ tasks |
| 5 | SP 800-60 Vol II | Custom OSCAL (euCann-developed) | 171 info types, C/I/A impact levels |
| 6 | AI RMF Playbook | nist_ai_rmf_playbook.json | Govern, Map, Measure, Manage functions |

Unifying layer: Same parsing pipeline handles all catalog-based frameworks. OSCAL model consistency makes this possible.

03

FRAMEWORK MODULES

SP 800-53 Rev 5 — Catalog Browser

The OSCAL Catalog as a UI

1 Family View

20 control families as tiles with counts per baseline

2 Baseline Filter

LOW / MODERATE / HIGH / PRIVACY from OSCAL profile include-controls

3 Control List

Sorted by ID; enhancements nested under parents

4 Control Detail

Statement, guidance, parameters, related controls, baselines, enhancements

5 Search & Favorites

Full-text index + user annotations in SQLite

6 Parameter Display

ODP labels inline with param-id identifiers and guidelines

Pro Features: Custom baseline creation, advanced multi-criteria filter dashboard, control comparison side-by-side view

NIST CSF 2.0

Cybersecurity Framework — Functions, Categories, Subcategories

GV GOVERN

4 Categories

Organizational context, risk strategy, roles, policy, oversight

ID IDENTIFY

3 Categories

Asset management, risk assessment, improvement

PR PROTECT

5 Categories

Identity management, awareness, data security, platform security, resilience

DE DETECT

2 Categories

Continuous monitoring, adverse event analysis

RS RESPOND

4 Categories

Incident management, analysis, reporting, mitigation

RC RECOVER

2 Categories

Recovery plan execution, communication

106 Subcategories browsable with full OSCAL catalog hierarchy. **Cross-mapped to SP 800-53 and SP 800-171 controls.**

SP 800-218 — Secure Software Development Framework

SDDF Practices, Tasks, and Implementation Examples

PO Prepare the Organization

5 practices

Define security requirements, implement roles and responsibilities, implement toolchains, define and use criteria for software security checks, create secure environments

PS Protect the Software

3 practices

Protect all code, verify third-party components, configure the compilation/build/packaging processes for security

PW Produce Well-Secured Software

8 practices

Design to meet security requirements, review designs, reuse secure code, create source code adhering to practices, test and verify code, configure and harden software

RV Respond to Vulnerabilities

3 practices

Identify and confirm vulnerabilities, assess and prioritize, remediate vulnerabilities

19 practices, 42+ tasks with implementation examples and references — all from the OSCAL catalog model.

AI RMF Playbook

AI Risk Management — Govern, Map, Measure, Manage

GOVERN

Policies, processes, procedures, and practices across the organization related to AI risk management

MAP

Context is established and understood, including interdependencies and potential impacts of AI system usage

MEASURE

Quantitative and qualitative methods to analyze, assess, benchmark, and monitor AI risk and related impacts

MANAGE

Risk resources are allocated to mitigate risks and maximize benefits of AI systems on a regular basis

Growing in relevance: Every organization building or deploying AI tools needs this framework. All subcategories and suggested actions browsable offline.

SP 800-60 Information Types

Automated FIPS 199 Categorization

Select Information Types

- C.2.8.12 Payment Transactions H / M / L
- D.6.1 System Development L / L / L
- C.2.3.2 Tax Collection M / M / L
- C.2.1.1 Federal Financial Mgmt M / M / L

↓ Auto-calculate

System: MODERATE

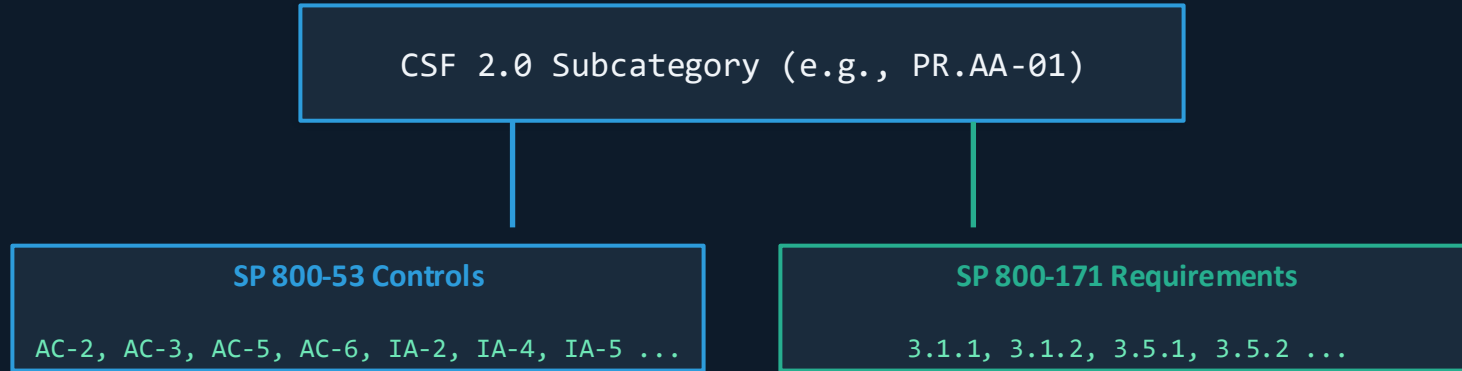
C: HIGH | I: MODERATE | A: LOW

- 171 information types from SP 800-60 Vol II, organized into 20 categories
- Each type carries NIST-recommended C/I/A impact levels
- FIPS 199 high-water mark: system level = highest per dimension
- Results written directly into the OSCAL SSP information-types array

Note: SP 800-60 is not yet available as official NIST OSCAL content. The OSCAL-formatted data was custom-developed by euCann. We submitted an RFC proposing a **Reference Taxonomy** model to standardize this. See OSCAL Discussion #2194.

Cross-Framework Mappings

Connecting Frameworks via OSCAL Mapping Data



Real-World Use Cases

- CSF assessment → which 800-53 MODERATE controls satisfy each outcome?
- 800-53 HIGH ATO → which 800-171 CUI requirements already covered?
- Building an SSP → CSF context for each control being implemented

04

OSCAL ARCHITECTURE

OSCAL at the Core — Data Architecture

Bundled OSCAL JSON Assets (~17MB)

- NIST_SP-800-53_rev5_catalog.json
- ..._HIGH/MODERATE/LOW/PRIVACY_profile.json
- NIST_CSF_v2.0_catalog.json
- NIST_SP800-171_rev3_catalog.json
- NIST_SP800-218_ver1_catalog.json
- csf_to_800_53/171_mappings.json
- llm_enhanced_assessment_objectives.json
- + SP 800-60, AI RMF, parameter blocks

~17MB

Runtime Pipeline

- 1 JSON assets loaded from bundle
- 2 Catalogs parsed → typed Dart models
- 3 Baseline profiles → control ID sets
- 4 Parameter placeholders extracted
- 5 Assessment objectives cross-referenced
- 6 Mappings linked at query time
- 7 Persisted to SQLite for annotations

OSCAL Model Hierarchy: Catalog → Groups → Controls → Params, Parts, Enhancements | Profile → Imports → Include-Controls | SSP → System-Characteristics, Control-Implementation

Architecture Overview

| | |
|-------------------------|--|
| Flutter UI Layer | iOS · Android · macOS · Windows · Linux · Web |
| Feature Modules | 800-53 · CSF · 800-171 · SSDF · AI RMF · 800-60 · SSP Builder · Assessment |
| Core Services | OscalService · AssessmentService · BaselineLoader · ParameterService · ExportService |
| State Management | Provider / ChangeNotifier · AppDataManager · ProjectDataManager |
| Data / Models | OscalCatalog · OscalControl · OscalParam · OscalProfile · InformationSystem |
| Local Storage | SQLite/SQLCipher (encrypted) · SharedPreferences · flutter_secure_storage |

Optional Cloud

TIMA RAG

Ollama · 6,425+ docs
Vector DB

Supabase Cloud

Auth · Profile sync
SSP backup

| Decision | Rationale |
|-----------------------------|---|
| Offline-first, bundled JSON | Air-gapped/regulated environments; zero-latency browsing |
| No custom data format | OSCAL files ship unmodified; app is consumer, not transformer |
| SQLCipher encryption | SSP content is sensitive; encrypted at rest on mobile |

05

SSP & ASSESSMENT

SSP Builder

BETA

10-Step Guided Wizard → OSCAL SSP JSON Export

| Step | Captures | OSCAL Location |
|-------------------|---------------------------------|---|
| 1. System Info | Name, description, auth type | <code>metadata, system-characteristics</code> |
| 2. Info Types | SP 800-60 types + FIPS 199 | <code>system-information.information-types</code> |
| 3. Impact Levels | C/I/A + auto high-water mark | <code>security-sensitivity-level</code> |
| 4. Auth Boundary | Boundary, network, data flows | <code>authorization-boundary</code> |
| 5. Roles | Owner, AO, ISO, ISSO, custom | <code>metadata.parties</code> |
| 6. Components | HW, SW, services inventory | <code>system-implementation.components</code> |
| 7. Controls | Baseline or custom selection | <code>import-profile</code> |
| 8. Implementation | Statements + param substitution | <code>implemented-requirements[]</code> |
| 9. Assessment | 800-53A objectives tracking | <code>Assessment annotations</code> |
| 10. Export | Completeness score + JSON | <code>system-security-plan root</code> |

Dual-View: Modern Wizard (guided) for first-time users, Classic Dashboard (direct access) for power users

Community ask: Review our responsible-parties coverage, import-profile href, information-type URIs, set-parameters completeness

800-53A Assessment Tracking

BETA

Embedded in the SSP Builder — Full SP 800-53A Rev 5.1.1

| Component | Implementation |
|-----------------------|--|
| Assessment Objectives | Structured per control, browsable and filterable |
| Methods (E/I/T) | Examine, Interview, Test tags on each objective |
| LLM Guidance | AI interpretation bundled offline per objective |
| Status Tracking | Met / Not Met / N/A — persisted per system |
| Param Substitution | ODP values from SSP rendered in assessment text |

1,193

controls with assessment
objectives covered

All 800-53A Rev 5.1.1
Examine / Interview / Test

Workflow

Browse
Control

View
Objectives

Review
Guidance

Mark
Status

Add
Notes

06

AI & ADOPTION

AI-Powered Features

| Feature | Description | Status |
|-------------------------|--|-----------------|
| LLM Assessment Guidance | AI interpretation for all 1,193 controls — bundled offline | ✔ Live |
| NISTBot Chat | Conversational NIST guidance with follow-up actions | 🔧 Built |
| TIMA Dialog RAG | 6,425+ NIST docs indexed, citations | 🔧 Code-complete |
| Contextual AI Buttons | "Ask AI" on every control detail screen | 🔧 Built |

Philosophy: AI as accelerator, not replacement.

1,193 AI-interpreted assessment objectives ship inside the binary. Zero backend dependency. Works in a SCIF.

LLM interprets NIST language and suggests evidence approaches. Practitioner makes all judgment calls.

Why This Matters for OSCAL

Every user of OSCAL Pocket Guide is consuming OSCAL — they just don't know it.

| Proof Point | Implication for OSCAL |
|--|---|
| Offline catalog browsing with full fidelity | OSCAL catalogs can serve as runtime databases |
| Profile-based filtering with zero custom logic | OSCAL profiles are naturally implementable |
| SSP generation from a guided wizard | Non-technical practitioners can produce machine-readable SSPs |
| Cross-framework navigation via mappings | OSCAL mapping publications unlock real workflows |

The Adoption Flywheel: More tools → more practitioners encounter OSCAL indirectly → more organizations produce OSCAL assets → more interoperability → more demand for OSCAL tools.

07

TECHNICAL DETAILS

Technical Details

Parameter Resolution · Baseline Filtering · OSCAL Versions

Parameter Resolution

```
{{ insert: param, ac-02_odp.01 }}
```

Scan statement for placeholder patterns via regex

Look up param by id → resolve user value or show label

Export as set-parameters on implemented-requirement

Baseline Filtering

Parse profile → Set<String> of control IDs

Filter control list against the set — O(1) swap

Handles explicit and implicit child inclusion

OSCAL Model Versions

| Catalog | Ver | Notes |
|-------------------|--------|--|
| SP 800-53 Rev 5 | 1.1.2 | Standard catalog with full parts hierarchy |
| SP 800-171 Rev 3 | 1.2.0 | Different schema — metadata fields diverge |
| SP 800-218 (SSDF) | 1.0.4 | Heavy use of nested parts for tasks |
| CSF 2.0 | Custom | Groups-of-groups model |

The 800-171 version difference (1.2.0 vs 1.1.x) was the most significant engineering challenge — requires runtime version detection.

By the Numbers

6

NIST Frameworks

Single app, single codebase

6

Target Platforms

iOS · Android · macOS · Win · Linux · Web

1,193

Controls Assessed

Full 800-53A Rev 5.1.1 coverage

6,425+

NIST Docs Indexed

TIMA RAG corpus

~17MB

Bundled OSCAL JSON

Offline-first, zero network

50K+

Lines of Dart

Flutter single codebase

08

ROADMAP & NEXT STEPS

Roadmap

Today (v1.5.1)

- ✓ 6-framework offline browsing
- ✓ SP 800-53 baseline profiles (4)
- ✓ Cross-framework mappings
- ✓ SP 800-60 + FIPS 199 auto-categorization
- ✓ LLM-enhanced guidance (bundled)
- ✓ Supabase authentication (code completed, feature not yet integrated)
- 🔧 SSP Builder + OSCAL export
- 🔧 800-53A assessment tracking

2026 Priorities

| Quarter | Feature |
|---------|---------------------------------------|
| Q1-Q2 | NISTBot + TIMA RAG production backend |
| Q2 | OSCAL SSP import (round-trip) |
| Q2 | Cloud sync across devices |
| Q3 | Reverse mappings (800-53 → CSF) |
| Q3 | Custom baselines + Web app |
| Q4 | OSCAL component definitions |
| 2027 | Assessment results + collaboration |

Long-term vision: A full OSCAL toolkit covering catalog, profile, component-definition, SSP, assessment-plan, assessment-results, and POAM models.

Get Involved

Try It. Break It. Tell Me What's Wrong.

iOS App Store: Search "OSCAL Pocket Guide" (ID: 6744904723)

Google Play: Search "OSCAL Pocket Guide"

1 OSCAL SSP Export Review

responsible-parties, import-profile href, information-type URIs, set-parameters

2 OSCAL Import Design

Loading existing SSPs, consuming from other tools, importing profile overlays

3 Model Coverage Gaps

Component definitions? Assessment plans/results? Catalog authoring?

4 Beta Feedback Program

SSP Builder, NISTBot chat, cloud sync — join the beta

No code contribution required — honest practitioner feedback is the most valuable thing you can give.

Questions?

OSCAL Pocket Guide

Making OSCAL accessible, mobile, and actionable.

iOS App Store:

apps.apple.com/app/id6744904723

Google Play: Search "OSCAL Pocket Guide"

This app would not exist without the NIST OSCAL team's work publishing high-quality, open OSCALJSON.